Estudio integración sistema de mensajería instantánea en plataforma comunicaciones unificadas UPCcom.

PFC-ETSETB Ingeniería en Electrónica

José Luis Villacampa

Agradecimientos:

Este trabajo está dedicado a Cristina y Mónica, mi hija y mi mujer. A ambas les agradezco la paciencia que han tenido durante estos meses y sobre todo a mi hija, quien me ha dejado cambiar biberones por un poco de tiempo para poder tirar adelante este trabajo.

Quiero agradecer la ayuda prestada por mi director de proyecto y por mi ponente, José Antonio y Enric. Ha sido indispensable vuestra colaboración y ayuda.

Muchas gracias por la ayuda y confianza prestada por SIETCG, especialmente a Albert. Al final no han ocurrido incidentes en vuestra red.

A UPCnet, por los recursos facilitados y por la confianza depositada en mí.

Índice

1. Introducción

1.1 Marco del Proyecto:

Durante los últimos años las comunicaciones telefónicas han ido evolucionando, dando un giro tecnológico y conceptual.

Hasta mediados de los años 90, la telefonía privada de las empresas estaba basada en sistemas convencionales, sistemas telefónicos digitales. Fue entre 1995 y 1997, cuando los fabricantes empezaron a trabajar en nuevos protocolos que permitían la codificación de pequeños fragmentos de voz en paquetes IP.

Entre 1998 y 1999, en una primera fase de la evolución de la telefonía convencional hacia la telefonía IP, aparecen las primeras centrales telefónicas con el hardware que permite codificar la voz en paquetes IP, de esta forma se aprovechaban los enlaces de datos entre delegaciones de una empresa para la transmisión de la voz, e incluso para hacer off-net, es decir, a través de los enlaces de datos se envían aquellas llamadas que no son para los usuarios telefónicos de una centralita ubicada en una región concreta, sino que se envían las llamadas nacionales con destino el de la región donde se ubica esta delegación, de esta forma las llamadas salen a la red pública desde esta centralita, convirtiéndose en llamadas locales y por tanto, ahorrando costes.

Sobre 2004 se inicia la segunda fase de esta evolución, aparecen los primeros sistemas puramente de telefonía IP, estos sistemas además de permitir la interconexión entre delegaciones mediante enlaces de voz, permiten la conexión de terminales IP. El uso de estos terminales IP elimina el cableado dedicado a la telefonía, ya que se conectan a la misma red de datos que los ordenadores de la organización y además, facilitan la movilidad, el teléfono se puede trasladar y conectar en otra mesa u otra planta sin necesidad de rehacer cableados o reprogramar el terminal.

A principios de 2005, arranca la tercera fase de la evolución de la telefonía, los sistemas telefónicos se integran con las aplicaciones. A este nuevo concepto se le conoce como comunicaciones unificadas.
En los comienzos de la evolución de la tecnología de VoIP, se creía que el gran beneficio era sólo la reducción de costes. En realidad este es un gran beneficio, pero no es ni el único, ni el más importante. El gran beneficio de esta tecnología viene de la mano de las nuevas funcionalidades de valor añadido que aporta.

Un sistema de comunicaciones unificadas, esta basado en el control de presencia. Por lo tanto, la comunicación esta centrada en la persona, no en el terminal. Cuando se contacta con alguien, se tendrá la seguridad de haber contactado con la persona deseada.
Este control de presencia, permite controlar la disponibilidad de tus contactos, de esta forma se puede elegir con quien y como nos vamos a comunicar.
La integración con aplicaciones como la mensajería instantánea de la organización, e-mail, transferencia de ficheros, aplicaciones propias del negocio, etc, aumenta la eficiencia de las comunicaciones.

En el siguiente esquema, se resumen los diferentes conceptos que se integran bajo una plataforma de comunicaciones unificadas:

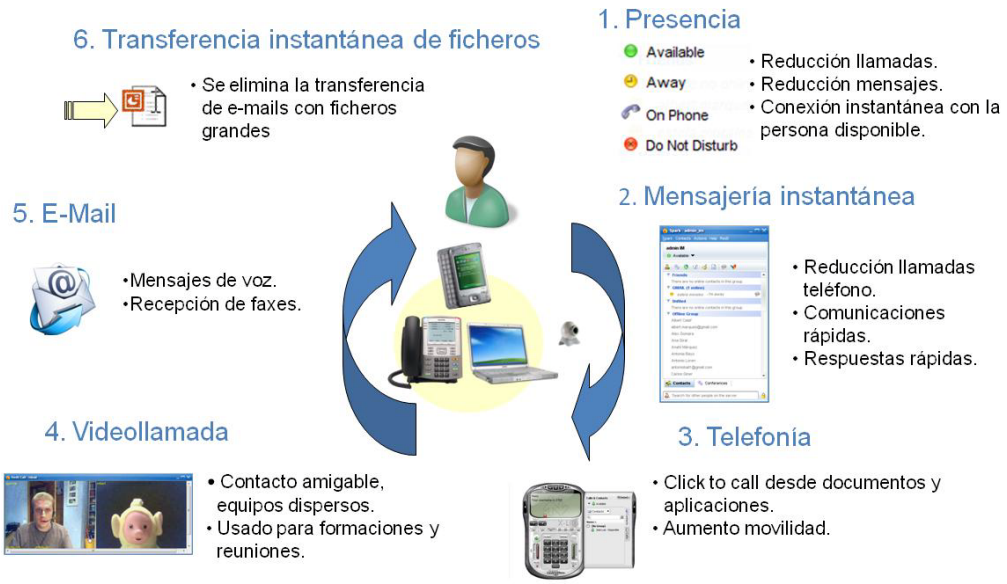


Figura 1. Servicios integrados bajo una plataforma de Comunicaciones Unificadas.

En los últimos años, en el colectivo universitario cada vez se ha hecho más patente la necesidad de integrar servicios de telefonía**,** e-mail, video conferencia y mensajería instantánea y hacerlos converger hacía una misma plataforma de comunicaciones unificadas.

La UPC de la mano de UPCnet y bajo el proyecto UPCcom, ha iniciado un ambicioso proyecto de implantación de un sistema de comunicaciones unificadas basado en telefonía IP. Este proyecto esta dividido en tres fases:

En una primera fase ya abordada, se han integrado los nuevos sistemas de telefonía IP con los antiguos sistemas de telefonía tradicional mediante enlaces digitales de voz, utilizando estos últimos como gateways entre la red pública y los usuarios de la telefonía tradicional.

En una segunda fase, se pretende dotar a la nueva plataforma de telefonía de nuevas funcionalidades de valor añadido, tales como la integración con una plataforma de mensajería instantánea, control de presencia, videollamada, transmisión de escritorio y transferencia de ficheros.
Será en esta segunda fase donde se centrará nuestro trabajo, y será en esta fase donde se dote de las principales funcionalidades de valor añadido, que permitirán convertir la plataforma de voz IP en una plataforma de comunicaciones unificadas.

En una tercera fase final, toda la infraestructura de voz convergerá a la nueva plataforma de telefonía IP, dotando de enlaces IP y RDSI a la nueva plataforma, desmantelando todas las plataformas de telefonía tradicional, y migrando todos los usuarios de telefonía tradicional a la nueva plataforma.

Una de las directrices del proyecto UPCcom, es potenciar el uso de software libre en la plataforma de comunicaciones unificadas. Además de reducir costes en el uso de los enlaces de datos para la transmisión de las llamadas de voz, se pretende reducir costes en hardware y en licencias software que acarrea cualquier tecnología ligada a fabricante.

Gracias a la nueva plataforma de comunicaciones unificadas, la UPC podrá fomentar el tele trabajo, ya que por fin se dispondrán de las herramientas necesarias para poder trabajar desde casa como si se estuviera en la oficina.

1.2 Descripción de las necesidades actuales:

Hay estudios que demuestran que en una jornada laboral de 40 horas semanales, cada empleado dedica 30 minutos en buscar los números de teléfono a los que va a llamar[1]. Si a esto además le sumamos que probablemente cuando se intenta localizar a alguien no se va a poder contactar con él, tenemos como resultado una reducción de la eficiencia en las comunicaciones humanas no debida a factores técnicos.

En los últimos años junto con la proliferación de aplicaciones de mensajería instantánea gratuitas tales como: Skype, MSN Messenger, Google talk, Yahoo Messenger, etc... muchas empresas han descubierto como este tipo de aplicaciones les están ofreciendo una forma rápida y eficiente de comunicarse con la persona deseada, en el momento necesario y en tiempo real. Sin embargo un uso de este tipo de aplicaciones de forma no controlada y de forma abierta a toda la red de Internet, en lugar de presentarse como una ventaja puede presentarse como un inconveniente, es decir las aplicaciones de mensajería instantánea se pueden usar para contactar con un miembro de un departamento homólogo de otro campus, para discutir algún ensayo o para concretar una reunión de trabajo, pero también pueden ser usadas para comentar con los amigos el último estreno de la cartelera cinematográfica o para planear el fin de semana. Si el uso que se va hacer no es un uso estrictamente laboral, la eficiencia de estos empleados disminuye aún mucho más.

La realidad es que haciendo un uso responsable de estas aplicaciones, se presentan grandes ventajas:

- Aumenta movilidad; permite contactar con compañeros de una organización desde cualquier parte del mundo.

- Mejora la productividad; tus contactos serán accesibles desde aplicaciones, tales como la mensajería instantánea, correo o incluso desde el directorio corporativo.

- Mejora la efectividad de las comunicaciones; ejercer un control sobre la presencia de tus contactos permite conocer la disponibilidad de estos y por tanto, se evita perder tiempo intentando contactar con contactos que no están disponibles para atender llamadas o responder a mensajes instantáneos.

- Reduce costes de comunicaciones; además de reducir costes en las llamadas entre campus, una plataforma de comunicaciones unificada permite reducir costes en conferencias cuando se esta de viaje, los usuarios de esta plataforma podrán realizar llamadas a través de las aplicaciones software integradas en los PCs y cursando llamadas a través de las plataformas de VoIP corporativas o establecer contactos vía mensajería instantánea.

Respecto a la mensajería instantánea, dado que puede hacerse un uso irresponsable de este tipo de aplicaciones, a las empresas se les ha planteado la difícil elección de bloquear el uso de aplicaciones de IM[2], pero evaluando las numerosas ventajas que presentan y lo que suponen estas ventajas para una organización y más aún si se presentan integradas con las plataformas de voz IP, en la mayoría de casos no se ha optado por esta elección, sino que se ha optado por la opción de adoptar medidas para la gestión de su uso, buscando la manera de supervisar y controlarlas.

Además de lo anterior, una plataforma de mensajería instantánea integrada en la plataforma de voz IP presenta numerosas características por las que desde UPCnet se ha pensado en la mensajería instantánea como uno de los valores añadidos a sumar en su plataforma de comunicaciones unificadas.

---

[1] Estudio realizado por la compañía de investigación Harris corporation en junio de 2006.
[2] IM, Acrónimo inglés de Instant Messanging o MI, Mensajería Instantánea en castellano.

Algunas de estas características se presentan a continuación:

- Un mensaje instantáneo no es un simple mensaje de texto plano, en un mensaje instantáneo se pueden anexar fotos, documentos de office o cualquier otro tipo de documento que se anexaría en un e-mail.

- Cuando se necesita la interacción rápida con otros usuarios, teniendo un servicio de mensajería instantánea el servicio de e-mail queda relegado a un segundo plano, ya que cuando envías un e-mail no tienes la seguridad que el destinatario este conectado y pendiente de la bandeja de entrada de su correo y aunque leyese el correo, entre que lo lee, responde, el servidor de correo procesa el mensaje y llega al destinatario, ralentizaría mucho las comunicaciones. Por lo tanto, para la UPC la facilidad de una rápida interacción es muy valiosa.

- Debido a que las conversaciones son instantáneas, la posibilidad de que surja algún mal entendido es bastante baja. Con los correos electrónicos los malentendidos pueden persistir por un tiempo, al menos hasta que alguien tenga la oportunidad de explicarse.

- Aún trabajando desde casa o desplazado fuera de la oficina, puedes disponer de la mensajería instantánea como si estuvieras en la oficina e interactuar con tus compañeros.

- Algunas de las aplicaciones de mensajería instantánea son integrables con los servidores de telefonía IP de la UPC y también permiten la implementación de vídeo llamadas, chats, salas de conferencia, etc.

El punto de partida es una red de datos con tecnología Gigabit Ethernet, distribuida por 8 campus. Según los informes generados por las herramientas que monitorizan estos enlaces, la ocupación del ancho de banda, solo con tráfico de datos, esta por debajo del 20% en hora punta, esto nos garantiza las mínimas necesidades de ancho de banda para la puesta en macha de un servicio de mensajería instantánea integrado con las plataformas de voz IP.

Actualmente hay aproximadamente 5000 usuarios de telefonía, con lo que se prevé la implantación del servicio de IM también para unos 5000 usuarios, cifra que puede variar considerablemente si en algún momento determinado se decidiera ofrecer este servicio también a los estudiantes.

1.3 Características de la plataforma a implementar:

La plataforma de mensajería instantánea a implementar en la UPC deberá cumplir con las siguientes características:

- **Autenticación:** La principal tarea de una plataforma de mensajería instantánea es la de verificar los credenciales de los usuarios. Una buena plataforma es capaz de interactuar con terceros sistemas, tales como la plataforma de gestión de usuarios de la que dispone la UPC, de tal forma que la gestión de usuarios se haga de forma simple y centralizada.

- **Seguridad:** Igual que con cualquier otro sistema de comunicaciones, la seguridad es un punto muy importante a tener en cuenta y tanto o más con las comunicaciones de mensajería instantánea. La plataforma que se implemente en la UPC deberá garantizar la seguridad en las comunicaciones instantáneas.

- **Registro conversaciones:** Probablemente algún tipo de norma o ley nos obligue en algún momento a registrar conversaciones o monitorizar las conversaciones y alertar cuando se hace un mal uso de la mensajería instantánea. Por lo tanto, esta plataforma deberá ser capaz de reconocer palabras clave, almacenar conversaciones y que el acceso a estas conversaciones sea fácilmente accesible.

- **Extensible**: La plataforma de mensajería instantánea partirá de una serie de características estándar, pero deberá permitir la posibilidad de ciertas integraciones mediante plugins que ampliarán el abanico de características.

- **Administración:** La plataforma de mensajería instantánea deberá ser fácilmente administrable.

- **Escalable:** Inicialmente esta previsto que se dará servicio de mensajería instantánea a unos 5000 usuarios, pero esta cifra fácilmente pude ampliarse por lo que la plataforma a implementar deberá ser fácilmente escalable para dar servicio a todos los usuarios que lo necesiten.

- **Integrar IM con usuarios de Softphones:** Habrán usuarios que para facilitar la movilidad, dispondrán de teléfonos software instalados en sus PCs. La plataforma de mensajería instantánea deberá ser capaz de integrarse con la mensajería instantánea que proporcionan los softphones y permitir el intercambio de mensajes IM entre ambas plataformas.

- **Integrar Presencia:** De la misma forma que deberá haber un flujo de mensajes de IM entre softphones y clientes de la plataforma de IM, ambos deberán registrar la presencia y estado, tanto de usuarios telefónicos como de clientes de IM.

- **Plataforma robusta:** La plataforma a implementar deberá ser robusta ante posibles fallos, hardware y software. Se deberá diseñar un sistema de alta disponibilidad para evitar cualquier pérdida de servicio.

- **Adaptable:** Deberá ser adaptable a las necesidades de la UPC, tanto a las necesidades docentes como funcionales de la administración y PAS.

- **Licenciamiento libre:** Siguiendo las directrices del proyecto UPCcom, las licencias software deberán ser gratuitas.

2. Objetivos

Para la definición de los objetivos de este proyecto se han mantenido reuniones con el Project Manager de UPCnet, responsable del proyecto UPCcom y con algunos miembros de su equipo.

Los objetivos definidos son los siguientes:

El hito principal de este proyecto es dotar a la nueva plataforma de comunicaciones unificadas de la UPC de un sistema de mensajería instantánea. Este servicio de valor añadido se sumará al nuevo servicio de VoIP, del que se dispondrá en la UPC en breve. Se prevé dotar de esta nueva funcionalidad a los mismos usuarios que van disponer de servicios de voz basados en voz IP bajo la plataforma de UPCcom, la previsión inicial es de implantar mensajería instantánea a unos 5000 usuarios.

La nueva plataforma de mensajería instantánea deberá garantizar de alguna forma el servicio ante posibles problemas hardware o software.

Hoy en día son diversas las plataformas hardware y sistemas operativos distribuidos por toda la UPC, por lo que los clientes de mensajería instantánea deberán ser multiplataforma, deberán de poderse instalar sobre terminales con Windows, Linux, Mac OS X e incluso deberán permitir el acceso a la plataforma de IM vía web.

El servicio de mensajería instantánea deberá integrarse con la plataforma de voz IP de UPCcom, integrando el intercambio de mensajes instantáneos entre los clientes Voip-software con los clientes de IM y viceversa, integrará presencia entre los mismos clientes y entre los teléfonos de sobremesa. Los clientes de la nueva plataforma de mensajería instantánea, deberán de ser capaces de proveer de un cliente de telefonía software embebido bajo una misma aplicación y que este pueda registrarse en la plataforma de voz IP, de esta forma cada usuario de un teléfono de sobremesa tendrá duplicada su extensión en forma de supletorio telefónico en una aplicación software sobre su PC, lo que facilitará la movilidad de los usuarios ya que registrando su cliente IM desde el exterior dispondrá de los mismos servicios de comunicaciones de los que dispone en la propia oficina.

En el edificio Vertex, la UPC tiene un Contact Center con 20 agentes que actualmente están trabajando en campañas de recepción de llamadas bajo una plataforma de telefonía IP. A día de hoy no hay ninguna planificación sobre como quedará estructurado y que servicios prestará este Contact Center, pero la previsión final es la de dotarle de nuevas funcionalidades de valor añadido, por lo tanto, la nueva plataforma de mensajería instantánea deberá estar preparada para ofrecer servicios de atención web vía chat, de esta forma además de poder contactar con el Contact Center vía telefónica se podrá contactar vía web-chat desde un link que pueda estar habilitado en cualquier página web de la UPC.

En este trabajo pretende seleccionar la nueva plataforma de mensajería instantánea que se integrará con la voz IP de UPCcom y se hará un estudio sobre los diferentes servicios de valor añadido que se pueden introducir y que sería interesante disponer de ellos en la nueva plataforma de comunicaciones unificadas de la UPC.

Este estudio finalizará con la puesta en marcha de una plataforma piloto de pruebas con un grupo de usuarios reducido. El despliegue final dependerá de otros calendarios distintos al seguido en este proyecto.

3. Tecnologías implicadas

En este capítulo se van a evaluar las distintas opciones con las que se puede implementar el nuevo sistema de mensajería instantánea. Tras una revisión de las principales alternativas se seleccionará el sistema con el que se implementará la nueva plataforma de comunicaciones unificadas.

Además, previo a la fase de diseño de la nueva plataforma, es necesario conocer cómo funcionan los protocolos con los que vamos a trabajar.
Sobre la nueva plataforma hay una extensa pila de protocolos, pero serán dos los protocolos que debamos estudiar para poder desarrollar la máxima integrabilidad posible entre la plataforma de voz IP y la plataforma de mensajería instantánea. Estos protocolos serán SIP y XMPP o Jabber.
Por un lado nos encontraremos con los usuarios de VoIP que "hablaran" en SIP y por otro lado, la plataforma de mensajería instantánea "hablará" en XMPP. Estos conceptos previos serán de gran ayuda a la hora de establecer una pasarela entre ambos protocolos.

3.1 Elección de la plataforma de mensajería instantánea para la UPC:

Actualmente las plataformas de mensajería instantánea se pueden clasificar en tres grupos:

- **Plataformas abiertas a todo www:** Tales como Skype, MSN Messenger, Google talk, Yahoo Messenger, etc.

- **Plataformas No GPL[3]:** La plataforma No GPL que actualmente esta copando el mercado ligado a fabricante es la de Microsoft, con su Office Communications Server 2007 R2.

- **Plataformas GPL:** Las plataformas GPL actualmente disponibles en el mercado, implementan el protocolo Jabber, protocolo basado en el estándar XML y gestionado por XMPP Standards Foundation[4]. Jabber es un protocolo abierto y público.

Jabber además de ser un protocolo libre, tiene las siguientes características:

- **Protocolo abierto:** Con todas las ventajas del software libre, se puede programar un servidor o un cliente o ver el código, entre otras cosas.

- **Descentralizado:** Se puede crear un servidor para Jabber, y se puede ínter operar o unirse a otras redes Jabber.

- **Testeado:** Las primeras implementaciones fueron desarrolladas en 1988 y ahora son muy estables. Existen miles de servidores en Internet que utilizan este protocolo, y millones de personas utilizándolo en servicios públicos de mensajería instantánea, tan conocidos como Google Talk o en implementaciones privadas en organizaciones.

- **Extensible:** Se puede ampliar con mejoras sobre el protocolo original. Las extensiones comunes son gestionadas por la XMPP Standards Foundation.

[3] Acrónimo en Ingles de "General Public License". Licencia creada por la "Free Software Foundation" y que esta orientada a proteger la libre distribución, modificación y uso de software.
[4] XMPP Standards Foundation, http://xmpp.org/

- **Seguro:** Cualquier servidor Jabber puede estar aislado del exterior. Jabber soporta SSL para comunicaciones cliente-servidor y algunos clientes aceptan GPG como cifrado de las comunicaciones. Esta en desarrollo el uso de claves de sesión y SASL.

- **Multiredes:** Una pasarela permite comunicarse con otros protocolos usados por clientes como MSN Messenger, ICQ, AOL o Yahoo!.

- **Salas de conversación:** Conocido como Multi-User chat o salas de chat. Es una de las extensiones que han sido añadidas a la mensajería Jabber, la cual le permite la creación de grupos de debate, como en las redes IRC, con la posibilidad de poseer usuarios con distintos privilegios (moderadores, participantes e invitados), iniciar conversaciones privadas y transferir archivos.

Además de lo atractivas que puedan resultar todas las características del protocolo Jabber, la plataforma de comunicaciones SIP implementada por UPCnet es Openser y en concreto la versión 1.3.1. OpenSer tiene un módulo XMPP, este módulo hace de gateway entre Openser y un servidor Jabber, lo que nos permitirá el intercambio de mensajes instantáneos y presencia entre clientes software SIP y clientes Jabber.

Después de todo lo anterior, queda patente la necesidad de que el servidor corporativo de la UPC, implemente el protocolo Jabber; ahora solo faltará ver que plataforma de todas las que hay en el mercado puede satisfacer todos los servicios suplementarios que necesita la nueva plataforma de comunicaciones unificada de la UPC.
A continuación se revisan algunas de las plataformas que implementan protocolo Jabber:

Jabberd14:

Es un servidor Jabber implementado en C/C++. Jabber14 es la implementación original del protocolo Jabber.

Las características principales son las siguientes:

- Servidor Jabber implementado en C/C++.
- Soporte excelente para protocolos de seguridad y encriptación.
- Soporta otros protocolos, no solo el protocolo XMPP/Jabber.
- Cumple estrictamente las normas del protocolo Jabber.
- Personalizable e integrable en sitios Web.
- Gran comunidad de desarrolladores.

Jabber XCP:

Jabber XCP es la plataforma XMPP/Jabber comercial líder. Jabber XCP destaca por su escalabilidad, extensibilidad y soporte multi-protocolo.

Las características principales son las siguientes:

- Flexible.
- Extensible.
- Personalizable.
- Fiable y escalable.
- Seguro.
- Código abierto e ínter operable con otras plataformas.

Openfire:

Openfire es una poderosa plataforma de mensajería instantánea y servidor Chat que implementa el protocolo XMPP/Jabber. Openfire destaca por su simplicidad a la hora de instalarlo y administrarlo, también destaca su flexibilidad a la hora de personalización y integrabilidad con otras aplicaciones.

Las características principales son las siguientes:

- Servidor Jabber implementado en Java.
- Administración basada en un interface Web amigable.
- Características adicionales desarrolladas e integradas a través de plugins.
- Plataforma independiente desarrollada puramente en Java.
- Personalizable.
- Seguro (SSL/TLS).
- Abierto e ínter operable con otras plataformas XMPP.
- Permite el registro de usuarios SIP desde cliente Jabber.

Tigase:

Tigase un servidor XMPP/Jabber ligero y escalable, también escrito en Java.
Se puede utilizar como aplicación integrada dentro de otro sistema o como un sistema "standalone". El bajo consumo de recursos hace que sea una buena solución para las pequeñas instalaciones y la escalabilidad hace que sea también bueno para los despliegues de muy alta carga y un enorme número de usuarios, puede instalarse en tantas máquinas como sea necesario. Es muy modular y extensible.

Las características principales son las siguientes:

- Servidor Jabber implementado en Java.
- Código abierto y libre.
- Robusto y fiable.
- Seguro (SSL/TLS).
- Flexible.
- Extensible.
- Fácil de instalar y mantener.

OpenIM:

OpenIM es un servidor Jabber implementado en Java y de código abierto. El propósito principal de OpenIM es proporcionar un servidor de mensajería instantánea, simple y altamente eficiente con una alta modularidad del código fuente.

Las características principales son las siguientes:

- Gran estabilidad.
- La integración con LDAP o BD puede ser fácilmente realizada.
- La mayoría de las funcionalidades de mensajería instantánea están soportadas.
- Comunicación servidor a servidor.
- Seguro (SSL).
- Almacenamiento y registro de conversación, para estadísticas o supervisión.
- No permite salas de Chat.

Ejabberd:

Los objetivos de diseño son la estabilidad y alto rendimiento. Incluye muchos servicios: chat, publicación de suscripción, directorio de usuarios de Jabber, etc y varios métodos de gestión: interfaz Web y herramienta de línea de comando.

Las características principales son las siguientes:

- Incluye un módulo de chat.
- Código abierto.
- Administración web y por línea de comandos.
- Tolerancia a fallos, clusterizable, distribuible, flexible, muy probado y fiable.
- Seguro (SSL/TLS).
- Flexible, arquitectura modular.

A continuación presentamos una tabla comparativa con las características más relevantes que son de interés para nuestra solución de mensajería instantánea:

	Jabberd14	Jabber XCP	Openfire	Tigase	Open IM	Ejabberd
Open Source	✓	✗	✓	✓	✗	✓
Autenticación	✓	✓	✓	✓	✓	✓
Seguridad	✓	✓	✓	✓	✓	✓
Extensible	✓	✓	✓	✓	✓	✓
Administración Web	✗	✗	✓	✗	✗	✓
Escalable	✓	✓	✓	✓	✗	✓
Gestión Presencia	✓	✓	✓	✓	✓	✓
Robustez	✓	✓	✓	✓	✓	✓
Cliente SIP integrado	✗	✗	✓	✗	✗	✗

Openfire cumple con todos los requisitos que inicialmente se plantearon. Además Openfire a través de sus plugins ofrece la posibilidad de poder registrar un usuario SIP en la plataforma VoIP de Openser, por lo tanto esto permitirá que un usuario tenga bajo un mismo cliente el cliente de mensajería instantánea y el cliente SIP con el que podrá hacer llamadas a otros usuarios SIP de la misma plataforma e incluso de la red conmutada.

Por lo tanto, será Openfire el servidor de mensajería instantánea seleccionado para proveer de estos servicios a la UPC.

3.2 Conceptos previos, Protocolo SIP:

El Protocolo SIP es un protocolo abierto que define una arquitectura de señalización y control para VoIP. Este protocolo fue desarrollado por la IETF[5] y con la colaboración de algunos líderes de las comunicaciones VoIP.

El protocolo SIP esta definido bajo la RFC[6] 3261, publicada en junio de 2002 y que sustituye a la antigua RFC 2543, que data de marzo de 1999.

[5] IETF, Internet Engieneering Task Force.

El propósito de SIP es la comunicación entre dispositivos multimedia. SIP hace posible esta comunicación gracias a dos protocolos que son RTP/RTCP y SDP.
El protocolo RTP se usa para transportar los datos de voz en tiempo real y el protocolo SDP se usa para la negociación de las capacidades de los participantes, tipo de codificación, etc.

SIP fue diseñado de acuerdo al modelo de Internet. Es un protocolo de señalización extremo a extremo que implica que toda la lógica es almacenada en los dispositivos finales. El estado de la conexión es también almacenado en los dispositivos finales. El precio a pagar por esta capacidad de distribución y su gran escalabilidad es una sobrecarga en la cabecera de los mensajes, debida a tener que mandar toda la información entre los dispositivos finales.

SIP es un protocolo de señalización a nivel de aplicación para gestionar el establecimiento y la gestión de sesiones con múltiples participantes. Se basa en mensajes de petición y respuesta y reutiliza muchos conceptos de estándares anteriores como HTTP y SMTP.

3.2.1 Arquitectura SIP:

SIP soporta funcionalidades para el establecimiento y finalización de las sesiones multimedia: localización, disponibilidad, utilización de recursos, y características de negociación.

Para implementar estas funcionalidades, existen varios componentes distintos en SIP. Existen dos elementos fundamentales, los agentes de usuario (UA) y los servidores SIP.

1. User Agent (UA): consisten en dos partes distintas, el User Agent Client (UAC) y el User Agent Server (UAS). Un UAC es una entidad lógica que genera peticiones SIP y recibe respuestas a esas peticiones. Un UAS es una entidad lógica que genera respuestas a las peticiones SIP.

   Ambos se encuentran en todos los agentes de usuario, así permiten la comunicación entre diferentes agentes de usuario mediante comunicaciones de tipo cliente-servidor.

2. Los servidores SIP pueden ser de tres tipos:
   - Proxy Server: retransmiten solicitudes y deciden a qué otro servidor deben remitirlas, alterando los campos de la solicitud en caso necesario. Es una entidad intermedia que actúa como cliente y servidor con el propósito de establecer llamadas entre los usuarios. Este servidor tienen una funcionalidad semejante a la de un Proxy HTTP que tiene la tarea de encaminar las peticiones que recibe de otras entidades más próximas al destinatario.

     Existen dos tipos de Proxy Servers: Statefull Proxy y Stateless Proxy.

     Statefull Proxy: mantienen el estado de las transacciones durante el procesamiento de las peticiones. Permite división de una petición en varias (forking), con la finalidad de la localización en paralelo de la llamada y obtener la mejor respuesta para enviarla al usuario que realizó la llamada.

     Stateless Proxy: no mantienen el estado de las transacciones durante el procesamiento de las peticiones, únicamente reenvían mensajes.
   - Registrar Server: es un servidor que acepta peticiones de registro de los usuarios y guarda la información de estas peticiones para suministrar un servicio de localización y traducción de direcciones en el dominio que controla.
   - Redirect Server: es un servidor que genera respuestas de redirección a las peticiones que recibe. Este servidor reencamina las peticiones hacia el próximo servidor.

[6] RFC, Request for Comments.

La división de estos servidores es conceptual, cualquiera de ellos puede estar físicamente en una única máquina, la división de éstos puede ser por motivos de escalabilidad y rendimiento.

3.2.2 Mensajes SIP:

SIP es un protocolo textual que usa una semántica semejante a la del protocolo HTTP. Los UAC realizan las peticiones y los UAS retornan respuestas a las peticiones de los clientes. SIP define la comunicación a través de dos tipos de mensajes. Las solicitudes (métodos) y las respuestas (códigos de estado) emplean el formato de mensaje genérico establecido en el RFC 2822 , que consiste en una línea inicial seguida de un o más campos de cabecera (headers), una línea vacía que indica el final de las cabeceras, y por último, el cuerpo del mensaje que es opcional.

Métodos SIP:

Las peticiones SIP son caracterizadas por la línea inicial del mensaje, llamada Request-Line, que contiene el nombre del método, el identificador del destinatario de la petición (Request-URI) y la versión del protocolo SIP.

Existen seis métodos básicos SIP (definidos en RFC 2543) que describen las peticiones de los clientes:

- INVITE: Permite invitar un usuario o servicio para participar en una sesión o para modificar parámetros en una sesión ya existente.
- ACK: Confirma el establecimiento de una sesión.
- OPTION: Solicita información sobre las capacidades de un servidor.
- BYE: Indica la terminación de una sesión.
- CANCEL: Cancela una petición pendiente.
- REGISTER: Registrar al User Agent.

Sin embargo, existen otros métodos adicionales que pueden ser utilizados, publicados en otros RFCs como los métodos INFO, SUBSCRIBER,etc.

A continuación se muestra un ejemplo real de mensaje del método REGISTER:

Figura 2. Mensaje método REGISTER.

Respuestas SIP:

Después de la recepción e interpretación del mensaje de solicitud SIP, el receptor del mismo responde con un mensaje. Este mensaje, es similar al anterior, difiriendo en la línea inicial, la llamada Status-Line, que contiene la versión de SIP, el código de la respuesta (Status–Code) y una pequeña descripción (Reason-Phrase). El código de la respuesta está compuesto por tres dígitos que permiten clasificar los diferentes tipos existentes.
El primer dígito define la clase de la respuesta:

1xx - Mensajes provisionales.
2xx - Respuestas de éxito.
3xx - Respuestas de redirección.
4xx - Respuestas de fallo de método.
5xx - Respuestas de fallos de servidor.
6xx - Respuestas de fallos globales.

A continuación se muestra un ejemplo real de un código de respuesta:

Figura 3. Ejemplo código respuesta.

A continuación, de manera más detallada explicando la causa concreta del error, se muestran las respuestas de error que se pueden producir en los mensajes SIP.

Como se ha indicado anteriormente corresponde con las respuestas de la clase:

4xx - Respuestas de fallo de método.
5xx - Respuestas de fallos de servidor.
6xx - Respuestas de fallos globales.

Estos errores se corresponden con los mensajes de error Q.931 o DSS1 y suponen el mapeo de los eventos SIP con los códigos de error de la RTC (Red telefónica conmutada).

Evento SIP	Detalle
400 Bad request.	Interworking, unspecified.
401 Unauthorized.	Bearer capability not authorized.
402 Payment required.	Call rejected.
403 Forbidden.	Bearer capability not authorized.
404 Not found.	Unallocated (unassigned) number.
405 Method not allowed.	Interworking, unspecified.
406 Not acceptable.	Interworking, unspecified.
407 Proxy authentication required.	Call rejected.
408 Request timeout.	Recover on Expires timeout.
409 Conflict.	Temporary failure.
410 Gone.	Unallocated (unassigned) number.
411 Length required.	Interworking, unspecified.
413 Request entity too long.	Interworking, unspecified.
414 Request URI (URL) too long.	Interworking, unspecified.
415 Unsupported media type.	Service or option not available.
420 Bad extension.	Interworking, unspecified.
480 Temporarily unavailable.	No user response.
481 Call leg does not exist.	Interworking, unspecified.
482 Loop detected.	Interworking, unspecified.
483 Too many hops.	Interworking, unspecified.
484 Address incomplete.	Address incomplete (invalid number format).
485 Address ambiguous.	Unallocated (unassigned) number.
486 Busy here.	User busy.
487 Request cancelled.	Interworking, unspecified.
488 Not acceptable here.	Interworking, unspecified.
500 Internal server error.	Temporary failure.
501 Not implemented.	Service or option not implemented.
502 Bad gateway.	Network out of order.
503 Service unavailable.	Service or option unavailable.
504 Gateway timeout.	Recover on Expires timeout.
505 Version not implemented.	Interworking, unspecified.
580 Precondition Failed.	Resource unavailable, unspecified.
600 Busy everywhere.	User busy.
603 Decline.	Call rejected.
604 Does not exist anywhere.	Unallocated (unassigned) number.
606 Not acceptable.	Bearer capability not presently available.

Figura 4. Respuestas de error en mensajes SIP.

3.2.3 Cabeceras SIP:

Las cabeceras se utilizan para transportar información necesaria a las entidades SIP. A continuación se detallan los campos:

- VIA: Indica el transporte usado para el envío e identifica la ruta del request, por ello cada proxy añade una línea a este campo.
- FROM: Indica la dirección del origen de la petición.
- TO: Indica la dirección del destinatario de la petición.
- CALL-ID: Identificador único para cada llamada y contiene la dirección del host. Debe ser igual para todos los mensajes dentro de una transacción.
- CSEQ: Se inicia con un número aleatorio e identifica de forma secuencial cada petición.
- CONTACT: Contiene una (o más) direcciones que pueden ser usada para contactar con el usuario.
- USER AGENT: Contiene información del cliente agente que realiza la comunicación.

A continuación se muestra un ejemplo real de cabecera SIP:

Figura 5. Ejemplo cabecera SIP.

3.2.4 Direccionamiento SIP:

Una de las funciones de los servidores SIP es la localización de los usuarios y resolución de nombres. Normalmente, el agente de usuario no conoce la dirección IP del destinatario de la llamada, sino su e-mail.

Las entidades SIP identifican a un usuario con las SIP URI (Uniform Resource Identifiers) definido en el RFC 2396. Una SIP URI tiene un formato similar al del e-mail, consta de un usuario y un dominio delimitado por una @, como muestra los siguientes casos:

usuario@dominio, donde dominio es un nombre de dominio completo.
usuario@equipo, donde equipo es el nombre de la máquina.
usuario@dirección_ip, donde dirección_ip es la dirección IP del dispositivo.
número_teléfono@gateway, donde el gateway permite acceder al número de teléfono a través de la red telefónica pública.

La solución de identificación de SIP, también puede ser basada en DNS, descrito en el RFC 3263, donde se describen los procedimientos DNS utilizados por los clientes para traducir una SIP URI en una dirección IP, puerta y protocolo de transporte utilizado, o por los servidores para retornar una respuesta al cliente en caso de que la petición falle.

3.2.5 Ejemplo llamada SIP:

A continuación se analizará detalladamente una llamada SIP. En una llamada hay varias transacciones SIP. Una transacción SIP se realiza mediante un intercambio de mensajes entre un cliente y un servidor. Consta de varias peticiones y respuestas y para agruparlas en la misma transacción esta el parámetro CSeq.

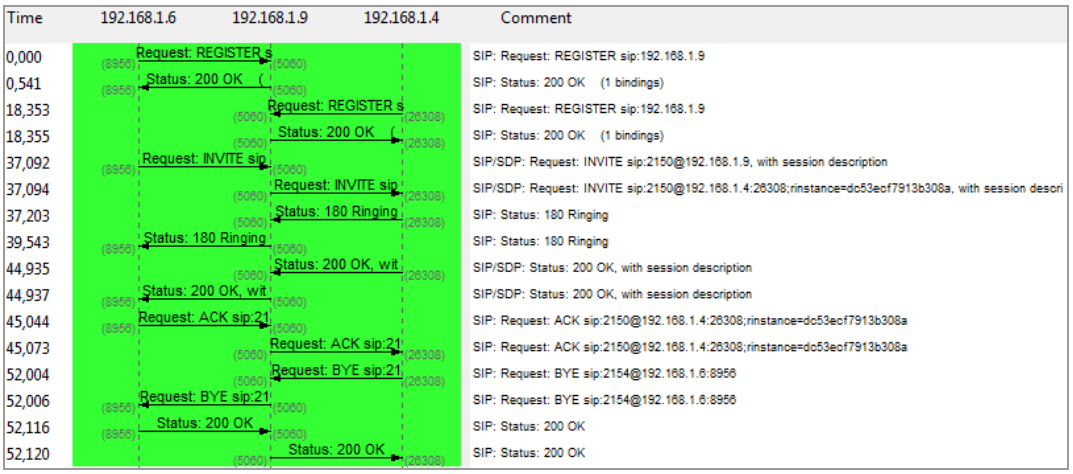


Figura 6. Ejemplo llamada SIP.

La dirección IP 192.168.1.6 corresponde al Usuario A, la dirección IP 192.168.1.4 corresponde al Usuario B y la dirección IP 192.168.1.9 corresponde al servidor proxy SIP, que en este caso esta dentro del mismo segmento de red que los usuarios A y B.

Las dos primeras transacciones corresponden al registro de los usuarios. Los usuarios deben registrarse para poder ser encontrados por otros usuarios. En este caso, los terminales envían una petición REGISTER, donde los campos from y to corresponden al usuario registrado. El servidor Proxy, que actúa como Register, consulta si el usuario puede ser autenticado y envía un mensaje de OK en caso positivo.

La siguiente transacción corresponde a un establecimiento de sesión. Esta sesión consiste en una petición INVITE del usuario A al proxy. Inmediatamente, el proxy reenvía la petición al usuario B. El usuario B envía un Ringing 180 cuando el teléfono empieza a sonar y también es reenviado por el proxy hacia el usuario A. Por ultimo, el OK 200 corresponde a aceptar la llamada (el usuario B descuelga).

En este momento la llamada está establecida, pasa a funcionar el protocolo de transporte RTP con los parámetros (puertos, direcciones, codecs, etc.) establecidos en la negociación mediante el protocolo SDP.

La última transacción corresponde a una finalización de sesión. Esta finalización se lleva a cabo con una única petición BYE enviada al Proxy, y posteriormente reenviada al usuario B. Este usuario contesta con un OK 200 para confirmar que se ha recibido el mensaje final correctamente.

3.3 Conceptos previos, Protocolo Jabbeer/XMPP:

Jabber es un protocolo abierto basado en el estándar XML para el intercambio en tiempo real de mensajes y presencia entre dos puntos en Internet. La principal aplicación de la tecnología Jabber es una plataforma extensible de mensajería y una red de MI (Mensajería Instantánea) que ofrece una funcionalidad similar a la de otros sistemas como AIM, ICQ, MSN Messenger y Yahoo.

Jabber destaca porque es distinto:

- Es **abierto**: el protocolo de Jabber es gratuito, abierto, público y comprensible. Además, existen múltiples implementaciones de código abierto para Servidores Jabber, como numerosos clientes y librerías de desarrollo.

- Es **libre**: Jabber es Libre porque no solo se puede ver cómo funciona, sino además el usuario tiene la libertad de implementarlo él mismo, la libertad de adaptarlo a sus necesidades, sin necesitar la aprobación de nadie.

- Es **extensible**: usando el potencial del lenguaje XML, cualquiera puede extender el protocolo de Jabber para una funcionalidad personalizada. Claro que para mantener la interoperatibilidad, las extensiones comunes son controladas por la Jabber Software Foundation.

- Es **descentralizado**: cualquiera puede montar su propio servidor de Jabber, además está libre de patentes y no depende de ninguna empresa de modo que se puede usar ahora y siempre con total libertad.

- Es **seguro**: Cualquier servidor de Jabber puede ser aislado de la red pública Jabber, cualquier implementación del servidor usa SSL para las comunicaciones cliente-servidor y numerosos clientes soportan PGP[7] y GPG[8] para encriptar las comunicaciones de cliente a cliente. Además, está en desarrollo una seguridad más robusta gracias al uso de SASL y contraseñas de sesión.

Los orígenes de Jabber datan de enero de 1999, fecha en la que Jeremi Miller publicó Jabber. Se trataba por aquel entonces de una tecnología abierta para la mensajería instantánea y la presencia en fase de desarrollo. Desde sus orígenes se han ido publicando varias RFCs, pero es en octubre de 2004, cuando la IETF publica finalmente el conjunto de RFCs que definen el protocolo XMPP actual.

En Febrero de 2002 la JSF[9] envió a la IETF un nuevo borrador (Internet Draft) del protocolo Jabber. Los resultados fueron prometedores y en Junio de ese mismo año se enviaron tres más. En todos estos borradores, la JSF cambió el nombre del protocolo por uno más neutral: eXtensible Messaging and Presence Protocol (XMPP).

Las RFCs que definen el actual protocolo XMPP son las siguientes:

- RFC 3920, define el núcleo del protocolo XMPP.

- RFC 3921, define los servicios de mensajería instantánea y de presencia previstos en XMPP.

Además, la IETF publicó dos RFCs complementarias:

- RFC 3922, que define una transposición de XMPP a CPIM (RFC 3860, Common Profile for Instant Messaging);

- RFC 3923, que define un mecanismo extremo a extremo de firma y cifrado de los objetos.

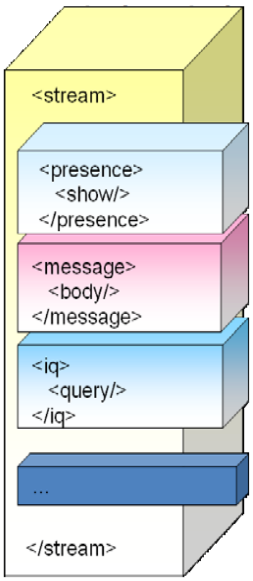
[7] PGP, Pretty Good Privacy. Herramienta de seguridad criptográfica que permite intercambiar archivos o mensajes con privacidad, autenticidad y conveniencia.

[8] GPG, GNU Privacy Guard. Implementación libre de PGP, licenciada bajo la GPL.

[9] JSF, Jabber Software Foundation.

3.3.1 Arquitectura XMPP:

El núcleo del protocolo XMPP esta compuesto por los siguientes componentes:



- **Stream:** Es la unidad contenedora de una secuencia de mensajes XMPP. En un stream pueden venir una o varias stanzas XML o bloques de mensaje que pueden ser de diferentes tipos.
- **Message:** Es una stanza delimitada para el intercambio de mensajes de información, normalmente de texto.
- **IQ:** Es una stanza especial de señalización Jabber, por la cual el protocolo ofrece un sistema de petición/respuesta para señalización entre clientes y servidores y para delimitar las funcionalidades del protocolo.
- **XEP:** Son el conjunto de especificaciones que extienden las diferentes stanzas que el protocolo soporta. Por ejemplo, el bloque *<presence>* es un subconjunto de directivas IQ empleadas para el control de presencia. Es donde se definen las extensiones del protocolo tales como: seguridad, intercambio de ficheros, soporte de VoIP, presencia, etc.

Figura 7. Componentes protocolo XMPP.

Generalmente, XMPP se implementa y se usa como una arquitectura cliente-servidor, pero XMPP no fuerza a hacerlo así, puede emplearse XMPP para establecer una comunicación directa, de extremo a extremo (P2P), entre los clientes.

En cuanto a los protocolos de transporte que sustentan la comunicación en XMPP, normalmente tenemos las conexiones puras de TCP, aunque podrían emplearse otros protocolos como HTTP Polling o HTTP Binding (también llamado BOSH), que se basan en usos muy particulares del protocolo de aplicación HTTP.

Un ejemplo de red de mensajería instantánea se presenta en la siguiente ilustración:

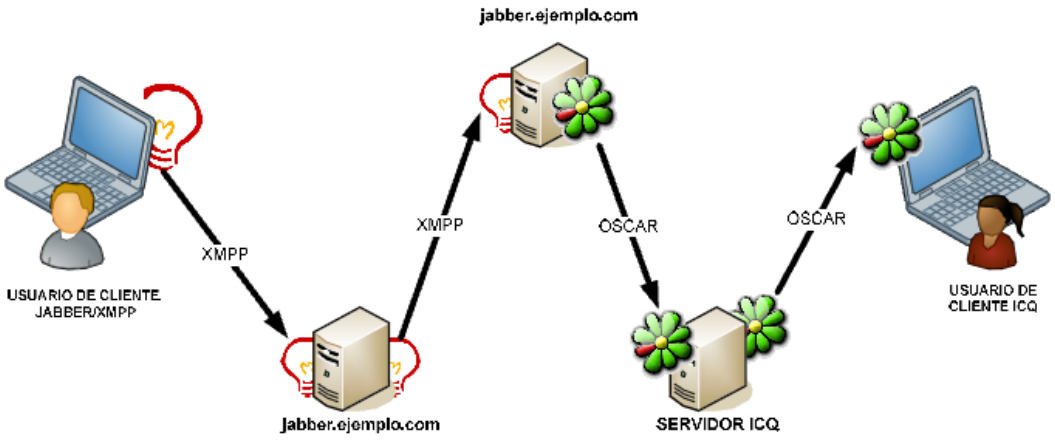


Figura 8. Ejemplo red IM.

OSCAR es el protocolo de mensajería instantánea utilizado en las redes ICQ y AIM.
La red puede tener múltiples clientes y servidores que se comuniquen por XMPP pero, además, puede contar con una serie de pasarelas o gateways que traducen de XMPP a otros protocolos

de IM de diferentes redes. Es posible, por tanto, comunicar usuarios de redes XMPP con redes de otros protocolos de MI, y viceversa, siempre que existan dichos gateways.

XMPP, además de gestionar el envío de paquetes XML hacia los clientes y/o hacia otros servidores de otras redes, también es capaz de intercambiar paquetes entre varios servidores XMPP. Para esta función, la IANA[10] tiene reservado el puerto TCP 5269.
Los clientes XMPP utilizan el puerto TCP 5222 para iniciar sesión.

3.3.2 Direccionamiento XMPP:

Para permitir el direccionamiento entre los clientes de las redes de mensajería instantánea, cada usuario de IM emplea un tipo de identificador que se denomina Uniform Resource Identifier (URI), que es una cadena de caracteres compacta y sujeta al formato *<user@host/resource>*, donde resource es opcional. También se emplean URIs con el formato *<chatroom@service>* y *<chatroom@service/nickname>*. Estas últimas se emplean para dirigirse a una sala de conversación de un determinado servicio de chat y para dirigirse a un participante de dicha conversación, respectivamente.
Las URIs también se denominan JIDs (Jabber IDs) por razones históricas. Los formatos que hemos comentado se basan en tres partes: un nodo, un dominio, y un recurso. Existen algunos límites en las longitudes de las partes de un JID: 1023 bytes por cada identificador de nodo, dominio, o recurso. Por tanto, las URIs o JIDs tendrán una longitud máxima de 3071 bytes, incluyendo los símbolos '@' y '/'. La definición de las URIs en notación BNF[11] es la siguiente:

```
jid = [ node "@" ] domain [ "/" resource ]
domain = fqdn / address-literal
fqdn = (sub-domain 1*("." sub-domain))
sub-domain = (internationalized domain label)
address-literal = IPv4address / IPv6address
```

De un JID, la única parte que debe declararse obligatoriamente es el dominio. Las otras dos son opcionales. La forma de especificar un dominio puede ser una de las siguientes: dirección IP completa o el nombre cualificado completo (FQDN). El identificador de nodo es opcional y, generalmente, representa al cliente, aunque también se puede usar para representar una sala de chat perteneciente a un servicio de multichat.
El identificador de recurso sirve para representar una sesión, una conexión, o un objeto.

3.3.3 Arquitectura de la comunicación basada en Streams y Stanzas:

El servidor participa en todas las comunicaciones XMPP. Su principal responsabilidad es la de proveer de servicios XMPP a los clientes de su dominio como pueden ser la redirección de sus paquetes y la gestión de sus cuentas de usuario.

El cliente XMPP es quien interactúa directamente con el usuario. Es el programa que muestra las respuestas del servidor y que recoge las peticiones del cliente y las envía al servidor para que las trate. Normalmente, estas peticiones son mensajes que el usuario quiere que lleguen a otro usuario, para lo que el servidor deberá localizar al destinatario y proceder a hacerle llegar el paquete por la ruta adecuada.

[10] IANA, Internet Assigned Numbers.
[11] BNF, Notación Backus-Naur form, notación utilizada en gramáticas de los lenguajes de programación, sistemas de comando y de los protocolos de comunicación.

La conexión se realiza entre cliente y servidor, y por ella viajan paquetes XML con las peticiones, mensajes, etc; que envían o reciben los usuarios en forma de fragmentos XML correctamente formados. El protocolo XMPP es quien especifica el formato de estos paquetes.
En el protocolo XMPP, toda la comunicación se hace a través de lo que denominan flujos XML o XML streams, representados por su sentencia XML <stream/>. Los XML streams son los contenedores XML de más alto nivel que permiten el intercambio de elementos XML entre dos hosts pertenecientes a una red de mensajería instantánea basada en protocolo XMPP. Son como un documento XML normal y corriente que tiene el tamaño tal que contiene toda la sesión, incluyendo cada mensaje enviado dentro de la sesión. Los streams no son documentos XML completos hasta que no se cierran dichas sesiones. El elemento raíz es <stream/>.
Cuando un cliente establece una conexión con el servidor (o con otro cliente) abre un XML stream. Entonces, el otro host abre otro stream de vuelta hacia el host llamante, de forma que cada conexión XMPP tiene dos XML streams abiertos, uno en cada dirección.
Los streams se emplean para enviar stanzas o comandos, de un host a otro. Las stanzas son los elementos hijos de primer nivel del documento XML del stream. Fundamentalmente, hay tres tipos principales de stanzas: <message/>, <presence/>
y <iq/>.

A continuación se muestra un ejemplo de mensaje XMPP stream. Este mensaje corresponde al intercambio de mensajes iniciales en el proceso de registro de un cliente XMPP en un servidor.

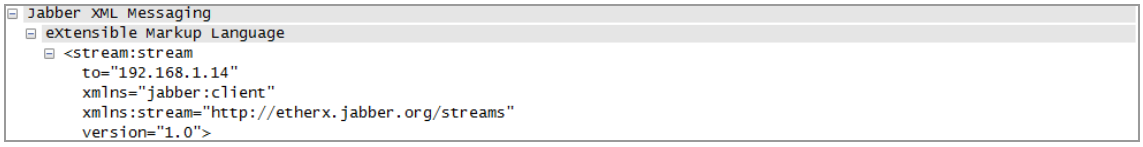


Figura 9. Ejemplo mensaje XMPP stream.

3.3.4 Seguridad en las comunicaciones XMPP:

Las comunicaciones XMPP entre clientes y servidores están securizadas principalmente a través de dos mecanismos: El protocolo TLS[12] aplicado a la capa de transporte y el protocolo SASL[13] aplicado a la capa de seguridad y autenticación simple.

Protocolo TLS:

El protocolo TLS es un protocolo para establecer una conexión segura entre un cliente y un servidor, o entre dos servidores. TLS es capaz de autenticar en ambos lados de la comunicación, y crea una conexión cifrada entre los dos. El protocolo TLS puede ser extendido, esto es que nuevos algoritmos pueden ser utilizados para cualquiera de los propósitos, con la condición de que tanto el cliente como el servidor conozcan dichos algoritmos.

La principal propiedad del protocolo TLS, es ofrecer privacidad e integridad de los datos, entre dos aplicaciones que se comunican; el protocolo está compuesto por dos capas, el TLS Record Protocol y el TLS Handshake Protocol.
El TLS Record Protocol ofrece seguridad en las conexiones y tiene dos propiedades básicas:

- La conexión es privada; se utiliza criptografía simétrica para el cifrado de los datos (DES, AES, RC4, etc), las llaves para los algoritmos simétricos son generadas una sola vez para cada sesión, y están basadas en un secreto negociado por otro protocolo (TLS Handshake), el TLS Record Protocol puede ser utilizado sin cifrado también.

[12] TLS, Transport Layer Security, protocolo definido en el RFC 2246.
[13] SASL, Simple Authentication and Security Layer, protocolo definido en el RFC 2222.

- La conexión es confiable, el transporte de mensajes, incluye una verificación de integridad de mensajes, utilizando una MAC con llave, para esto se utilizan algoritmos de funciones de hash seguros como SHA1, SHA256, MD5.

El TLS Record Protocol, es utilizado para la encapsulación de varios protocolos de nivel superior, uno de tales protocolos encapsulados, es el TLS Handshake Protocol, el cual es utilizado para autenticar tanto a los clientes como a los servidores, y para negociar un algoritmo de cifrado así como las llaves criptográficas, antes de que el protocolo de la aplicación transmita o reciba el primer byte de datos. El protocolo TLS Handshake Protocol ofrece seguridad en la conexión, y tiene 3 propiedades básicas:

- La identidad de la otra parte puede ser verificada utilizando criptografía asimétrica (RSA o DSS), esta autenticación puede ser opcional, pero generalmente se requiere por al menos una de las partes.

- La negociación de un secreto compartido es segura: el secreto negociado no está disponible para ningún atacante, incluso si el atacante utilizara un ataque hombre en el medio.

- La negociación es confiable: ningún atacante puede modificar la comunicación sin ser detectado por las partes que intervienen en la comunicación.

Si las conversaciones no estuvieran cifradas sería muy fácil un ataque tipo “man-in-the-middle” podría hacer que se interceptaran de forma fácil las conversaciones.
Este tipo de ataques son muy simples, con una herramienta de análisis de redes y protocolos como puede ser Wireshark se puede interceptar una conversación entre dos usuarios. Por ejemplo, en la siguiente imagen se puede ver como desde un equipo ubicado en la dirección IP 192.168.1.6 se están enviando mensajes hacia un equipo ubicado en la dirección IP 192.168.1.14:

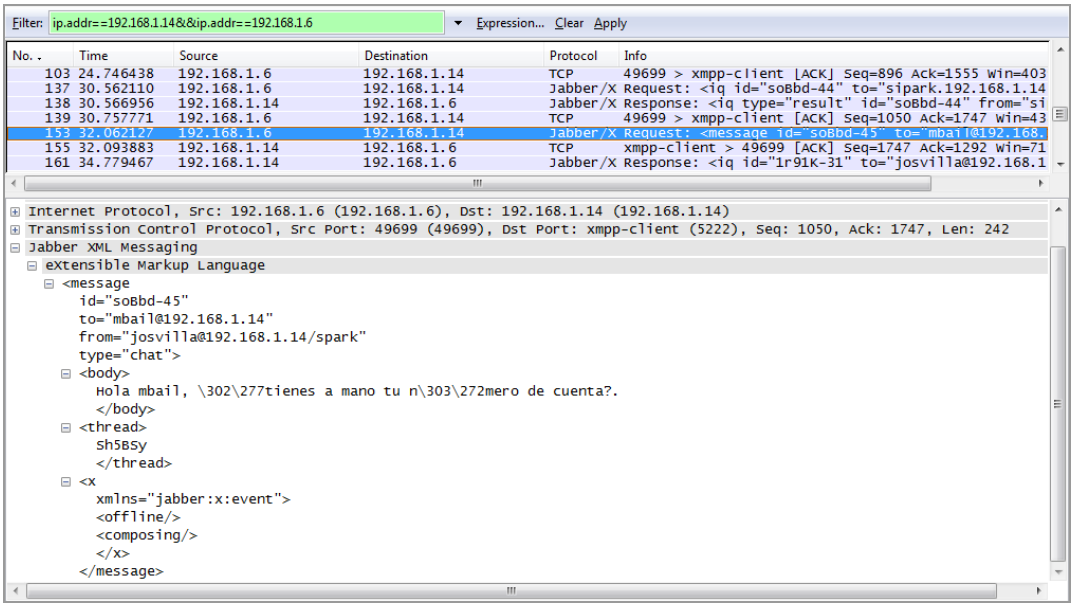


Figura 10. Ejemplo ataque “man in the middle”.

Analizando un poco más la captura realizada en el ataque se puede llegar fácilmente a una serie de conclusiones más:

1. Los usuarios implicados en la conversación son mbail y josvilla.
2. El usuario josvilla es el que envía el mensaje a mbail.
3. El usuario josvilla esta ubicado en la dirección IP 192.168.1.6.
4. El servidor Jabber en este caso esta ubicado en la dirección IP 192.168.1.14.

Si el servidor Jabber tiene habilitada la seguridad TLS, la captura de estos mensajes ya no es por lo menos tan fácil de conseguir.

En la siguiente imagen se puede ver una captura con Wireshark de una conversación cifrada con TLS:

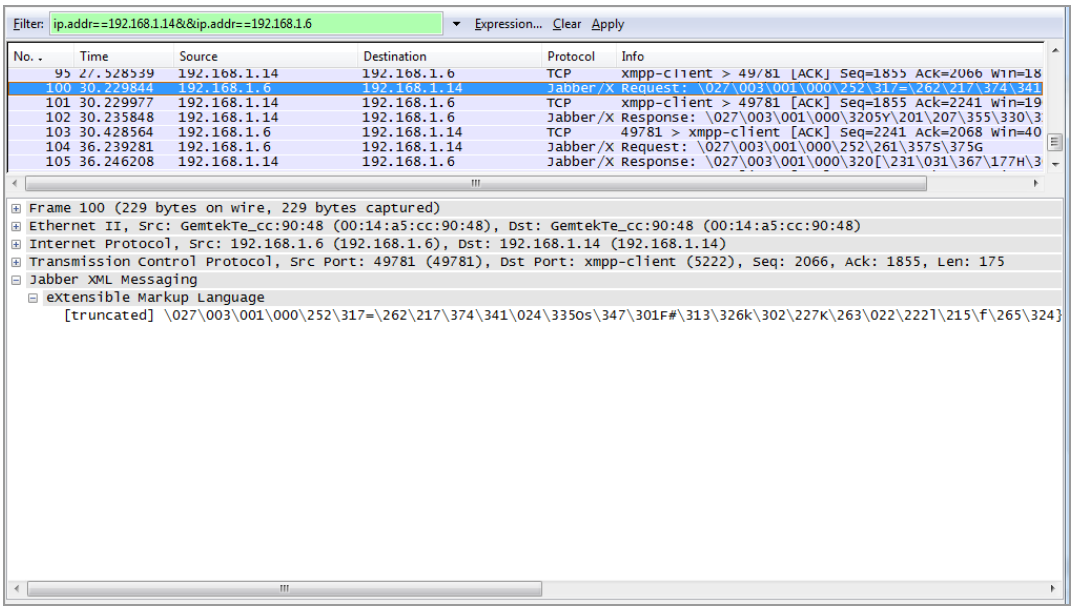


Figura 11. Captura tráfico securizado con TLS.

En este caso ya no se pueden interpretar los mensajes enviados, ni los usuarios implicados, ni las ubicaciones de los usuarios y del servidor.

Protocolo SASL:

XMPP incluye autenticación mediante SASL6 En los inicios del protocolo Jabber la autentificación se realizaba mediante el protocolo jabber:iq:auth, pero hoy en día esto se realiza empleando SASL.
SASL provee a XMPP de un método generalizado para la autenticación. Para ello se han aplicado ciertas normas:

- Si la autenticación SASL se da entre dos servidores, la comunicación no se establecerá hasta que cada servidor se asegure de la auténtica DNS del otro.
- Si quien quiere autenticarse soporta SASL, deberá incluir el atributo 'version' con el valor '1.0' por lo menos, en la cabecera del stream inicial.
- Si el servidor soporta SASL, deberá informar de sus tipos de autentificaciones con la etiqueta <mechanisms/> en la contestación de la etiqueta de inicio de sesión, si es que el cliente soporta la conexión SASL.
- Durante la negociación SASL, ninguno de los dos deberá enviar algún carácter en blanco como separación entre elementos, esta prohibición ayuda a asegurar la precisión a nivel de byte.
- Cualquier carácter XML contenido en los elementos XML deberá estar codificado usando el algoritmo base64[14].

[14] Base64, Algoritmo de codificación binaria de textos.

El proceso de autentificación mediante SASL sería el siguiente:

- La entidad que pida una autentificación SASL deberá incluir el atributo 'version' en la etiqueta de inicio de sesión enviada al servidor con el valor '1.0' como mínimo:

```
Jabber XML Messaging
  eXtensible Markup Language
    <stream:stream
        to="192.168.1.14"
        xmlns="jabber:client"
        xmlns:stream="http://etherx.jabber.org/streams"
        version="1.0">
```

Figura 12. Autentificación SASL, atributo versión.

- Cuando el servidor recibe la etiqueta de inicio de sesión con el atributo 'version' deberá comunicar los tipos de autentificación SASL que implementa, cada uno de ellos irá dentro de un hijo del tipo <mechanisms/>:

```
Jabber XML Messaging
  eXtensible Markup Language
    <stream:features>
      <starttls
          xmlns="urn:ietf:params:xml:ns:xmpp-tls">
        </starttls>
      <mechanisms
          xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
        <mechanism>
            DIGEST-MD5
          </mechanism>
        <mechanism>
            PLAIN
          </mechanism>
        <mechanism>
            ANONYMOUS
          </mechanism>
        <mechanism>
            CRAM-MD5
          </mechanism>
        </mechanisms>
      <compression
          xmlns="http://jabber.org/features/compress">
        <method>
            zlib
          </method>
        </compression>
      <auth
          xmlns="http://jabber.org/features/iq-auth"/>
      <register
          xmlns="http://jabber.org/features/iq-register"/>
      </stream:features>
```

Figura 13. Autentificación SASL, mechaninsms.

- El cliente deberá seleccionar uno de los mecanismos enviando el elemento <auth/> con un valor adecuado para el mecanismo de autentificación SASL elegido. Si el cliente debe responder con un elemento vacío, responderá con el carácter '=', que indicará que la respuesta no contiene datos:

```
Jabber XML Messaging
  eXtensible Markup Language
    <auth
        mechanism="PLAIN"
        xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
        AGpvc3ZpbGxhAGQ2YWdheg==
        </auth>
```

Figura 14. Autenticación SASL, auth.

- Si fuera necesario, el servidor enviará el elemento <challenge/> al cliente que contendrá datos en formato XML, esto dependerá del tipo de autentificación SASL que el cliente haya elegido.
- El cliente responderá al "desafío" enviando la etiqueta <response/> al servidor.

- Si fuera necesario el servidor enviaría más elementos <challenge/> y el cliente respondería a los mismos.

Esta serie de desafíos y respuestas continuaría hasta que ocurriera una de estas tres cosas:

- Que el cliente que quería autentificarse aborte la autentificación enviando la etiqueta <abort/> al servidor. En cuyo caso el servidor dejará al cliente enviar un número configurable de peticiones más, normalmente dos, antes de cerrar la conexión TCP. Así el cliente podrá volver a autentificarse sin necesidad de reiniciar la sesión como pasaba con el protocolo Jabber original.

- Que el servidor responda con la etiqueta <failure/>, con la que comunicaría al cliente que la autentificación ha fallado. Como en el caso anterior, le dejará enviar un limitado número de peticiones más para que, si lo desea, vuelva a intentarlo.

```
Jabber XML Messaging
  eXtensible Markup Language
    <failure
        xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
        <not-authorized/>
        </failure>
```

Figura 15. Protocolo Jabber, respuesta failure.

- Que el servidor responda con la etiqueta <success/>, con la que comunicaría al cliente que la autentificación se ha realizado correctamente, y además contendría datos en formato XML dependiendo del tipo de autentificación SASL. Una vez realizada la autentificación el cliente deberá enviar una etiqueta vacía de inicio de sesión, sin necesidad de cerrar la sesión anterior, a la que contestará el servidor y comenzará la conexión.

```
Jabber XML Messaging
  eXtensible Markup Language
    <success
        xmlns="urn:ietf:params:xml:ns:xmpp-sasl"/>
```

Figura 16. Protocolo Jabber, respuesta success.

En forma de resumen gráfico, podemos ver el proceso de autentificación SASL en la siguiente ilustración:

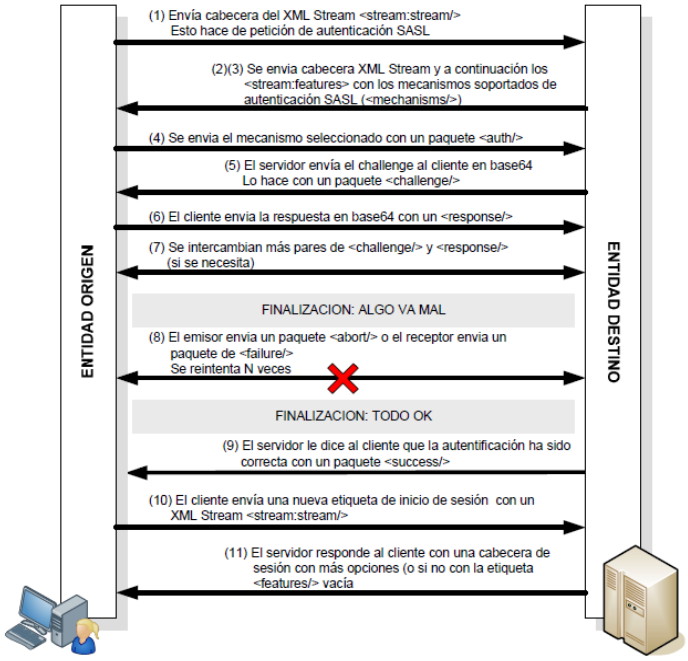


Figura 17. Proceso autenticación SASL.

3.3.5 Mensajería instantánea en XMPP:

Los mensajes son la parte más importante de cualquier sistema de mensajería instantánea. XMPP es un protocolo muy orientado a los mensajes, que pueden ser de seis tipos diferentes:

- **Normal:** que serían mensajes parecidos a los del correo electrónico.
- **Chat:** mensajes persona a persona que serían los mensajes utilizados en una conversación entre dos personas.
- **Groupchat:** mensajes enviados a un grupo de personas
- **Headline:** que serían los mensajes de marquesina
- **Error:** para los mensajes de error
- **Jabber:x:oob:** para las conexiones directas entre clientes para envío de archivos.

Los cinco primeros son los tipos más normales de mensajes en los sistemas XMPP. Los mensajes 'jabber:x:oob' se denominan out-of-band messages, y facilitan un mecanismo para intercambio de datos directamente entre dos usuarios, estos mensajes usan el servidor XMPP para intercambiar datos de la conexión entre los dos clientes, normalmente el usuario que va a servir un fichero enviaría un mensaje de este tipo al otro cliente con la IP y el puerto al que se debe conectar el cliente que va a descargarse el fichero. Se pueden enviar etiquetas de 'oob' dentro de la extensión 'x' de un mensaje normal, o empaquetadas dentro de un paquete del tipo Info/Query.

Direccionamiento de los mensajes:

El protocolo de mensajes es muy sencillo, los paquetes son enviados por un usuario a un receptor. Por defecto, no se reciben confirmaciones de que el mensaje ha sido recibido por el destinatario para reducir el tráfico en el servidor, además si el receptor no se encuentra disponible, el servidor guardará el mensaje hasta que se conecte. A este comportamiento particular del protocolo se le denomina "store and forward".

Cuando un usuario genera un mensaje, es responsabilidad de los servidores la entrega de dicho mensaje al destinatario. El destinatario puede estar conectado al mismo servidor que el remitente del mensaje, o no. En el último caso, el remitente necesita establecer una conexión con el otro servidor y entregar el mensaje, que tendrá que ser enrutado hacia el servidor destino.

El cliente de MI debería especificar el destinatario mediante su JID en el atributo to de la stanza <message/>:

Figura 18. JID del destinatario.

Si el mensaje es una réplica de otro mensaje recibido con anterioridad cuyo remitente tenía un JID determinado (con el formato user@domain/resource), entonces la réplica deberá dirigirse a la misma dirección, incluyendo el resource. Si el mensaje está siendo enviado fuera del contexto de un chat, la parte resource del JID puede obviarse (user@domain).

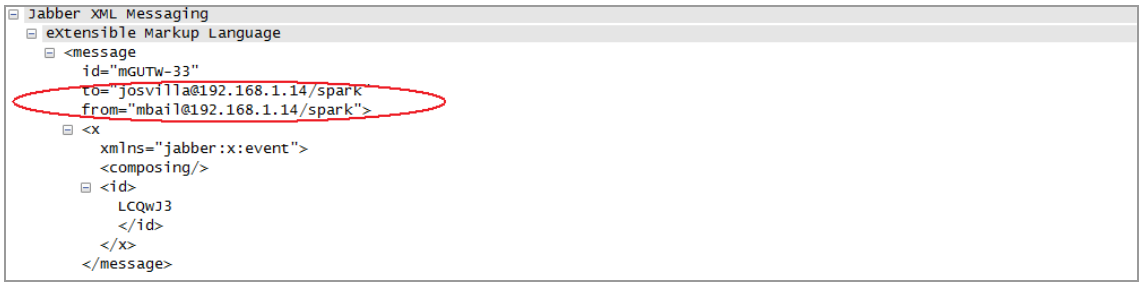


Figura 19. JID del remitente y destinatario.

Estructura de los mensajes:

Un mensaje básico consiste en un elemento <message/> con los atributos from, to e id. Además los elementos <message/> se componen de varios subelementos:

- **<subject>:** pensado para indicar el título o asunto relativo al mensaje.
- **<thread>:** para identificar la conversación a la que pertenece el mensaje.
- **<body>:** para especificar el contenido del mensaje.
- **<error>:** para especificar que ha ocurrido un error.

Un mensaje típico sería el siguiente:

Figura 20. Ejemplo mensaje Jabber.

De todas formas, muchos de estos campos no son obligatorios. Por ejemplo, el id y el <thread/> son para un manejo más fácil de los mensajes en los clientes, pero no todos los mensajes tienen por qué tener <subject/> y, además, el campo from no es necesario, ya que para evitar que una persona envíe mensajes poniendo como origen un Jabber ID que no es el suyo el servidor reescribirá el campo from del paquete antes de enviarlo a su destinatario.

También es importante señalar que los mensajes pueden contener más de un elemento de cuerpo del mensaje o <body/>, dependiendo del idioma utilizado, lo que se especifica con el atributo xml:lang.

El tipo de mensajes por defecto es el “Normal” y aquellos mensajes que no vienen especificados con el tipo, se consideran de tipo “Normal”.

En el caso de que la conversación tenga lugar en un contexto de chat, lo que significa que el atributo de tipo del mensaje es type='chat', el emisor de un mensaje es responsable de generar un valor de hilo y de copiar dicho valor en los sucesivos mensajes enviados dentro de la sesión, con el fin de vincular todos aquellos mensajes que son réplica del primero.

Los mensajes del tipo 'groupchat' son similares a los mensajes del tipo 'chat', pero están diseñados para mantener conversaciones entre un grupo de personas sobre un tema en concreto. Por ello, cuando un cliente envía un mensaje al grupo, todos los clientes que se hayan unido al grupo recibirán el mensaje. También se pueden escribir mensajes de tipo 'groupchat' de ámbito privado, dirigidos a un usuario en concreto.

Los mensajes del tipo 'headline' están pensados para enviar información de titulares de noticias para mostrar en la barra de estado o en otras partes de la interfaz de usuarios. Son una especie de servicio de news ticker. Los usan comúnmente los 'chatbots' para informar de noticias, alertas del tiempo… a los usuarios que se den de alta en estos servicios automatizados. No requieren del las etiquetas <thread/> o <subject/>.

Los mensajes de tipo 'error', que se envían Cuando un cliente envía un mensaje y se produce un error, ya sea que el cliente al que va dirigido no existe, o que sencillamente el mensaje ha sido rechazado por el cliente destinatario.

Por último están los mensajes de tipo 'out-of-band', que no son realmente un mensaje convencional de XMPP, sino una extensión especial de los mensajes que se envía dentro de un mensaje normal. Un mensaje de este tipo contiene información (normalmente una URL) que el cliente desea usar para realizar una transferencia de datos punto-a-punto, sin pasar por el servidor. Los clientes, normalmente, lo suelen implementar arrancando un servidor Web o un FTP separadamente o como parte del cliente Jabber/XMPP. Por lo tanto están dando a otro cliente la información de su IP y el puerto que han abierto para que el otro cliente se descargue el archivo.
Esto es ideal para el intercambio de ficheros entre clientes (P2P file transfer) ya que, con ello, consigues reducir el tráfico que pasa por el servidor. Sin embargo, como siempre que un cliente revela su IP tiene sus riesgos, si la persona a la que el usuario envía esta información es un usuario malintencionada podría utilizar la IP con otros fines muy distintos a los de descargarse el fichero que el cliente le está ofreciendo.

3.3.6 Presencia en XMPP:

XMPP se ha preocupado de la intimidad de los usuarios. Si un usuario quiere agregar a otro en su lista de contactos, esto implica recibir su presencia, por lo que debe solicitarlo a través del servidor. Los usuarios tienen derecho a la intimidad y serán ellos mismos quienes decidan quién puede conocer su estado actual y quién no.

El protocolo de presencia es usado principalmente en dos contextos:

- **Presence update:** actualización de la presencia debido a un cambio de estado del usuario.

- **Presence subscription management:** permite a los usuarios subscribirse a las actualizaciones de presencia de otros usuarios y controlar quien está accediendo a su propia presencia.

En ambos casos el servidor de XMPP actúa como árbitro entre el emisor de la actualización de presencia y los destinatarios de la misma. El servidor tiene la obligación de hacer llegar el paquete de actualización de presencia a todos los contactos del cliente que lo generó, pero sólo a los contactos que el cliente emisor ha confirmado que le gustaría que recibieran su presencia.

El servicio de actualización de presencia usa un simple mensaje unidireccional, del emisor al servidor de su dominio. El servidor tendrá que copiar y reenviar ese mensaje de presencia a todos los clientes del emisor. Para ello, el servidor puede consultar la lista de contactos del emisor para conocer quiénes son sus contactos. Esta lista de contactos en XMPP recibe el nombre de Roster.

Así, la lista de contactos (roster), se guarda en el servidor XMPP. La ventaja que tiene este modelo centralizado es mantener siempre los rosters actualizados, aunque se cambie de aplicación cliente XMPP o de ordenador.

Los mensajes de presencia viajan en stanzas <presence/>. Estas stanzas, a diferencia de las de tipo <message/>, no se dirigen a un usuario específico. Pueden incluir un atributo 'to', pero, generalmente, las stanzas <presence/> se envían al servidor, que es el que luego notificará la presencia a los contactos del cliente.

La presencia de XMPP funciona como un mecanismo de publicación-subscripción (publish/subscribe o PUB/SUB). Así, los clientes enviarán y recibirán mensajes de suscripción de presencia de dos tipos: 'subscribe' y 'unsubscribe'. Este tipo de suscripción de presencia también se emplea en los mensajes de grupo 'groupchat'.

Cuando cambia el estado de presencia en un cliente, este lo publica en el servidor para que este último lo notifique a los suscriptores de esa presencia.

Una vez que se establece una sesión con el servidor, el cliente XMPP debe enviar primero su información de presencia al servidor de forma que se comunica su disponibilidad para la comunicación. Esta notificación inicial de presencia permite al servidor enviar una prueba de disponibilidad a los usuarios suscritos a la presencia del cliente que acaba de conectarse.

Cambios en el estado de presencia de los clientes:

Si el usuario quiere cambiar su estatus, el proceso sigue unos sencillos pasos. Así, desde el cliente XMPP del usuario se genera una stanza <presence/> con toda la información necesaria.

En cualquier momento, durante la sesión, el cliente puede enviar una stanza de presencia al servidor sin especificar ningún atributo de tipo, de forma que funciona como un keep-alive o mensaje de notificación de que el usuario continúa conectado, u ocupado.

En el caso de recibir un mensaje de presencia, el servidor difunde el mismo mediante broadcasting a cada una de las entidades del roster del remitente cuyo atributo de tipo de suscripción es 'from' o 'both'. Más información de la disponibilidad de los usuarios se especifica mediante los elementos <status/> y <show/>.

Los elementos <status/> incluyen datos legibles expresados en lenguaje natural que describan el estado de la disponibilidad, que puede expresarse en varios idiomas tal y como permite el atributo 'xml:lang'.

Los elementos <show/> incluyen ciertos valores que expresan diferentes estados de disponibilidad de una entidad, están codificados para que puedan ser interpretados por el servidor.

Los clientes XMPP usarán normalmente el estado <show/> para mostrar iconos de presencia estándar, alertas sonoras y lanzar eventos. Si el estado <show/> no está indicado, el usuario se encuentra en estado normal u online. Los estados estándar para <show/> son:

- **chat:** el usuario está intentando hablar con alguien.
- **away:** el usuario está fuera del cliente XMPP por un corto periodo de tiempo.

- **xa (extended away):** el usuario está fuera por un periodo prolongado de tiempo.
- **dnd (do not disturb):** el usuario no desea recibir mensajes.

El protocolo permite la extensibilidad de estados <show/> codificados, ya que pueden definirse otros espacios de nombres para añadir otros subelementos a la stanza <presence/>.

Un ejemplo de mensaje de notificación de presencia podría ser el siguiente:

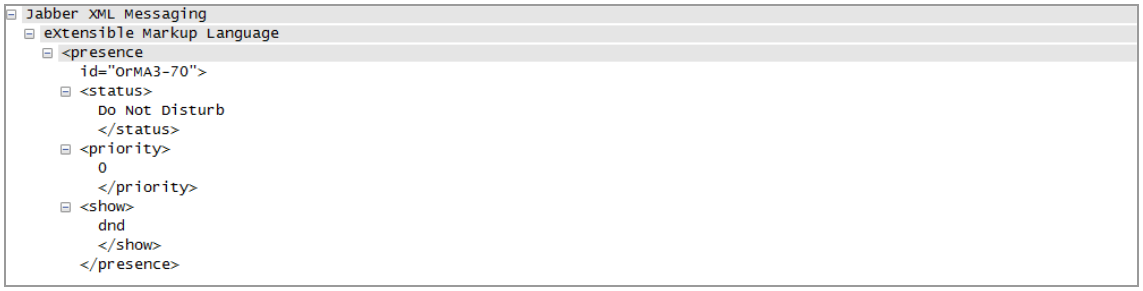


Figura 21. Ejemplo de notificación de presencia.

Administración del Roster:

La suscripción de presencia tiene la ventaja de reducir considerablemente el tráfico de red de un sistema de MI al no tener que hacer broadcasting o difusión masiva de los estados de presencia de los usuarios.

Para organizar y administrar las suscripciones de cada usuario, XMPP ha definido unas estructuras de datos estándar conocidas como roster. Un XMPP roster no es más que una lista de contactos identificados por su Jabber ID. El protocolo de suscripción de presencia permite a los usuarios suscribirse a la presencia de otros usuarios, sea cual sea su dominio en una red XMPP. Los servidores usan el protocolo de suscripción de presencia para sincronizar los rosters para sus usuarios tanto fuera como dentro de su
dominio.

Los diferentes tipos de relaciones de suscripción tratados por el roster son los siguientes:

- **to:** el usuario está interesado en recibir las actualizaciones de presencia del suscriptor.
- **from:** el suscriptor está interesado en recibir las actualizaciones de presencia del usuario.
- **both:** el usuario y el suscriptor tienen un mutuo interés en recibir las presencias entre ellos.
- **none:** ni el usuario ni el suscriptor tienen interés por recibir presencias entre ellos.

Además de la información básica de suscripción, el roster permite al usuario almacenar información estándar del interfaz de ese suscriptor. Esta información incluye un sobrenombre puesto por el usuario a su suscriptor y el grupo o grupos al que pertenece el suscriptor a la hora de mostrarlo en la interfaz.

Una de las cosas más importantes es que toda esta información está almacenada y administrada por el servidor. Esto simplifica notablemente la implementación y permite al usuario tener disponible dicha información allí donde se encuentre. Cualquier cambio
realizado en el roster en uno de los clientes será automáticamente actualizado en los demás clientes iniciados con el mismo Jabber ID. El protocolo roster fue desarrollado para permitir a los clientes XMPP la administración de sus contactos.

A pesar de la estrecha relación entre el roster y la presencia, son conceptos diferentes. Digamos que el roster es todo el conjunto de contactos relacionados con una cuenta Jabber o XMPP. Además podemos guardar datos de cada contacto como un sobrenombre puesto por el usuario o el grupo o grupos al que pertenece para poder así buscar todos los contactos de un grupo, y no tener que recorrernos todos los contactos. Como cada usuario tendrá su propio roster, si enviamos una actualización de presencia al servidor, éste buscará en nuestro roster todos los contactos que tengan los tipos de suscripción 'both' o 'from' para reenviarles sólo a ellos nuestra presencia.

Como todo se almacena en el servidor, el cliente comenzará nada más autentificarse pidiendo todo el roster al servidor con un 'roster get'. Mostrará entonces todos los contactos del roster en la aplicación. Si más adelante el usuario desea hacer cambios en algún contacto del roster enviará un 'roster update' al servidor y se quedará esperando un 'roster push' del servidor para confirmar la actualización a todos los clientes abiertos con el mismo Jabber ID, ya que, si uno de ellos realiza un cambio, éste se tiene que reflejar en el resto.

El protocolo roster es una extensión del protocolo IQ. Los tres tipos básicos de protocolos de gestión del roster son:

- **Roster get:** usado por los clientes para obtener una copia del roster almacenado en el servidor.

- **Roster update:** usado por los clientes para actualizar el roster almacenado en el servidor.

- **Roster push:** actualizaciones asíncronas del roster que el servidor envía a los clientes.

Los cambios del roster pueden suceder en cualquier momento desde que un usuario se autentifica, por lo tanto los clientes deben estar preparados para recibir del servidor 'roster push' y actualizar los contactos que muestran en ese momento. El servidor enviará un 'roster push' cuando:

- Una actualización del roster cambia los atributos del mismo, y se deba actualizar los clientes mostrados o alguno de sus atributos.

- Cuando por algún cambio en el tipo de suscripción de presencia se deba crear borrar un contacto.

Como se ha visto el protocolo de suscripción de presencia tiene grandes efectos sobre el roster, tantos que una actualización del tipo de suscripción puede hacer desaparecer de la pantalla del usuario un contacto, ya que si el contacto no desea ser visto por el usuario y envía una cancelación de la suscripción de presencia poniéndola a 'none' el usuario debe de dejar de ver el estado de ese contacto lo antes posible.

Los clientes sólo pueden modificar el apodo de sus contactos y el grupo o grupos a los que pertenecen mediante el protocolo roster. Además, como hemos visto pueden influir en el roster mediante el protocolo de suscripción de presencia.

El diseño de las suscripciones de presencia y los roster de XMPP facilitan el desarrollo de los clientes Jabber/XMPP, ya que los clientes no deben almacenar esos datos, ni tampoco se deben de preocupar de cómo modificarlos o administrarlos, lo único que deben hacer es realizar peticiones al servidor y será éste quien se encargue de su procesamiento. Además el servidor se debe encargar de que cuando el usuario se desconecte o conecte, todos sus suscriptores reciban una actualización de presencia.
Aun así los clientes son libres de enviar de por sí actualizaciones de presencia a otros usuarios. Sin embargo si esto se realiza, el cliente deberá gestionar que actualiza la presencia de todos sus contactos, mientras que si lo hace a través del roster, sólo le enviará la actualización de presencia al servidor y será éste quien lo gestione.

4. Desarrollo plataforma de comunicaciones unificadas:

En la fase de diseño de la nueva plataforma de mensajería instantánea de la UPC, se ha trabajado y desarrollado sobre la maqueta del laboratorio de pruebas. Para implementar la pasarela XMPP y el envío de presencia entre la plataforma de VoIP (Openser) y la plataforma de IM (Openfire), se requiere de varias paradas de ambas plataformas y estas paradas son inadmisibles para un entorno de producción.
A la hora de integrar la solución final se requerirá de una ventana de trabajo, fuera de horas para no interrumpir la producción.

En el diseño de la plataforma de mensajería instantánea de la UPC se ha trabajado en las siguientes líneas:

- Pasarela XMPP entre la plataforma de VoIP y la plataforma de IM.
- Traspaso de presencia entre la plataforma de VoIP y la plataforma de IM.
- Gestión usuarios a través de OpenLDAP.
- Integración usuarios SIP en clientes Jabber.
- Implementación videollamada desde Openfire.
- Implementación cliente Sparkweb.
- Implementación alta disponibilidad y alto rendimiento.
- Implementación servicios de web collaboration para centro de atención de contactos.
- Escalabilidad de Openfire.
- Pasarelas con otras plataformas de mensajería.

4.1 Descripción laboratorio de pruebas:

En la fase de análisis se ha decidido implementar un laboratorio de pruebas, entorno que nos permitirá implementar todos los objetivos marcados en este proyecto pero fuera del entorno real de producción.

El trabajar en un entorno de laboratorio de pruebas, en una primera instancia es debido a la seguridad implícita que nos introduce el trabajar fuera del entorno de producción, nos permitirá arrancar y parar cualquier servidor sin que esto repercuta a ningún usuario final, además podremos instalar aplicaciones sin ningún tipo de riesgo que implique la parada de ningún servicio crítico, como podría ser el servicio de VoIP, durante un tiempo indeterminado.
Otra ventaja colateral es que se puede trabajar desde cualquier lugar, sin problemas de accesos a edificios corporativos y CPDs, a cualquier hora del día y además nos permite administrar plataformas idénticas a las que tiene la UPC pero sin ningún tipo de restricción, esto último en un entorno real de producción no es posible y es básico para la integración de la plataforma de IM.
Otra de las ventajas que nos permite el laboratorio es, que nos permitirá ir el día de la instalación a uno de los CPDs de la UPC con el servidor de mensajería instantánea preparado y listo para ser puesto en marcha, integrándose con el servidor de VoIP y la plataforma de gestión de usuarios de la UPC. Esto reduce considerablemente el tiempo de trabajo desplazado a la UPC y la puesta en marcha es mucho más rápida, prácticamente se reduce al tiempo dedicado al arranque de la plataforma de IM (pocos minutos) y al tiempo dedicado a la validación de las pruebas en real.

4.1.1 Laboratorio virtual:

Una parte del entorno de laboratorio sobre el que se va a trabajar va a ser un entono virtual, sobre el que se virtualizarán las siguientes máquinas:

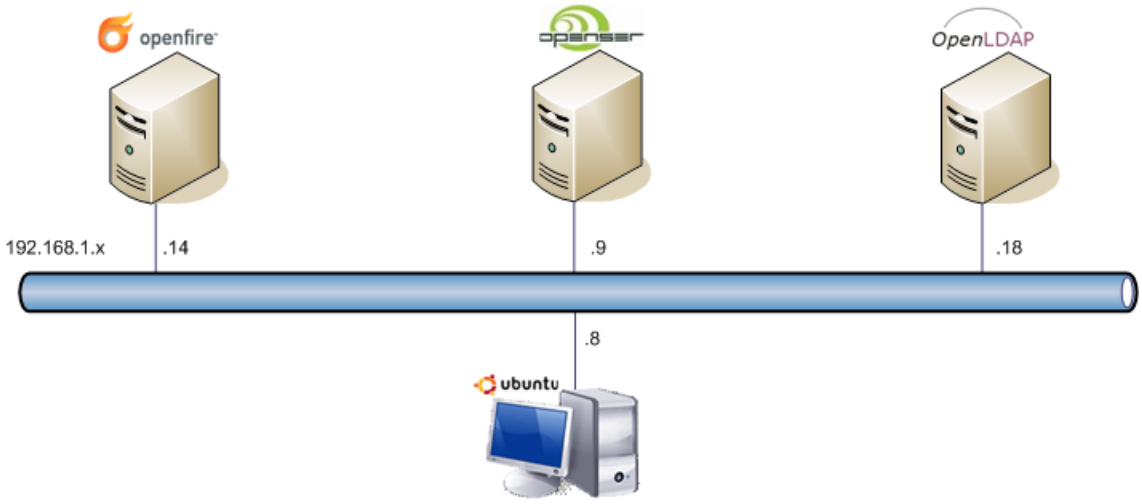


Figura 22. Esquema de red laboratorio virtual.

- **Openfire:** Es el servidor Jabber de mensajería instantánea.
- **Openser:** Es el servidor de VoIP. Se hará una instalación básica, con pocos usuarios y sin accesos a RTC. Nos permitirá trabajar la integración de la mensajería instantánea con la VoIP.
- **OpenLDAP:** Es el servidor de gestión de usuarios. Se hará una instalación básica, con pocos usuarios. Nos permitirá trabajar la integración de la mensajería instantánea con el servidor de gestión de usuarios.
- **Ubuntu:** Se trata de un Ubuntu Desktop, nos permitirá testear el cliente Desktop de Openfire sobre una plataforma Linux.

Para montar este laboratorio virtual son varias las herramientas software de virtualización que hay en el mercado, pero sin duda a día de hoy la herramienta líder del mercado por su solidez, estabilidad, seguridad y soporte de fabricante, es VMware.
VMware es altamente recomendado para entornos de pruebas, además hay dos versiones de VMware gratuitas con lo que se ajusta a las perspectivas económicas de este trabajo. Ambas versiones de VMware permiten la captura de Snapshots, cuando se captura un Snapshot se salva el estado de servidor en el momento de la captura, con las mismas aplicaciones y configuracion, de este modo en caso de que ocurra un desastre con una instalación es posible hacer “rollback” de forma rápida.

Las versiones VMware gratuitas son:

- **VMware ESXi:** Esta versión de VMware no corre sobre ningún sistema operativo, viene embebido con su propio sistema operativo, por lo tanto esto facilita el máximo rendimiento de las máquinas virtuales. Permite salvar múltiples Snapshots.
- **VMware Server:** Esta versión de VMware corre sobre los sistemas operativos Windows y Linux. Permite salvar el último Snapshots.

Arquitectura VMware ESXi y VMware Server:

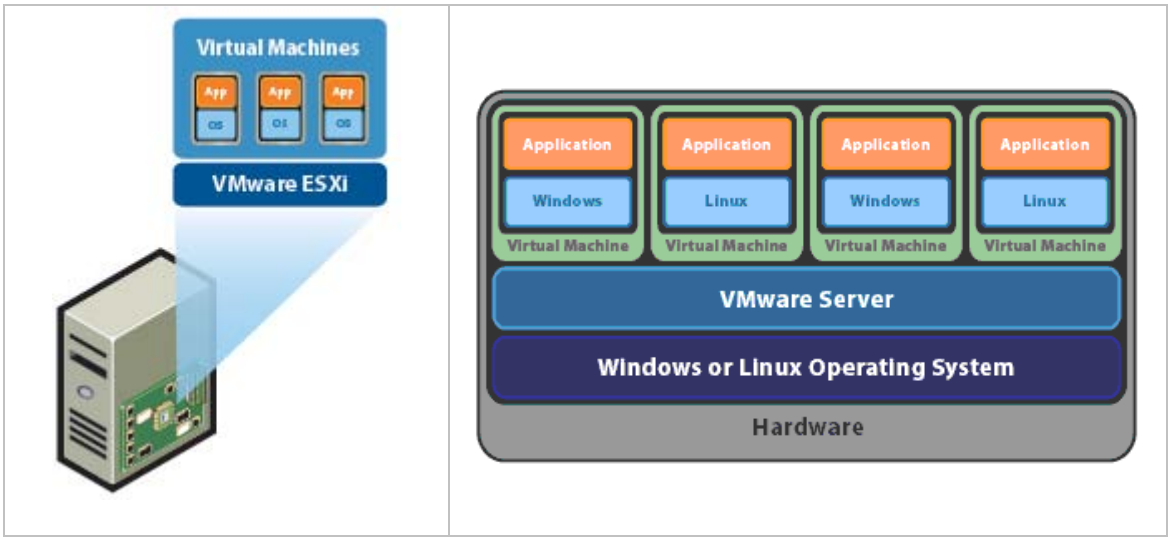


Figura 23. Arquitectura de VMware ESXi y VMware Server.

En las imágenes anteriores podemos ver las diferencias conceptuales entre la arquitectura de un VMware ESXi y un VMware Server.

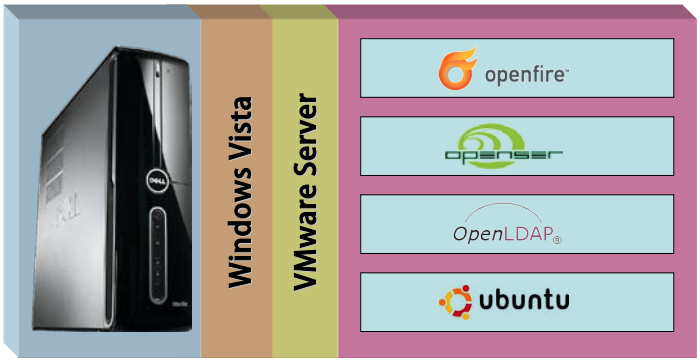
Para el de laboratorio de pruebas, puesto que se va a trabajar sobre un PC de uso doméstico con Windows Vista, la herramienta software de virtualización escogida es VMware Server.

VMware Server se instala sobre un PC de la marca DELL con las siguientes características:

Dell Studio Desktop Slim 540ST:

- Procesador Intel Core 2 Quad.
- 4 Gb de memoria RAM.
- 500 Gb de disco duro.
- Tarjeta gráfica 256 Mb ATI RADEON HD 3450.
- Tarjeta de red integrada Realtek, Gigabit.
- Tarjeta de red PCI 802.11b Asus.
- Lector/Grabador DVDs.

La arquitectura final del laboratorio virtual con el que vamos a trabajar esta representada en la siguiente imagen:



Finalmente en nuestro entorno de desarrollo, además de las máquinas virtuales descritas contaremos con el Windows Vista que tenemos instalado sobre el PC donde corre VMware y con otro PC Portátil que tiene instalado Windows XP.

La foto final del laboratorio de desarrollo es la siguiente:

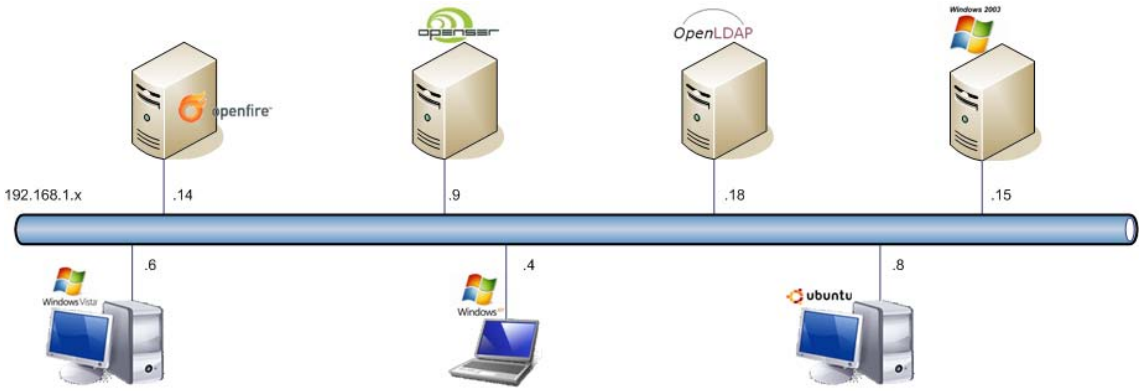


Figura 25. Esquema de red laboratorio de pruebas y estudio.

4.2 Desarrollo funcionalidades:

A continuación se describen los desarrollos realizados en las diferentes líneas estudiadas:

4.2.1 Pasarela XMPP entre las plataformas de VoIP e IM:

Establecer una pasarela XMPP entre Openser y Openfire nos permitirá el intercambio de mensajes instantáneos entre ambas plataformas.
Además de establecer la pasarela XMPP, para que se pueda producir el intercambio de mensajes instantáneos entre ambas plataformas, los usuarios de Openser deberán mapear a los usuarios de Openfire con la nomenclatura correcta y de forma idéntica ocurrirá con los usuarios Jabber de Openfire, que deberán mapear correctamente a los usuarios SIP de Openser.

La pasarela XMPP sigue el siguiente esquema:

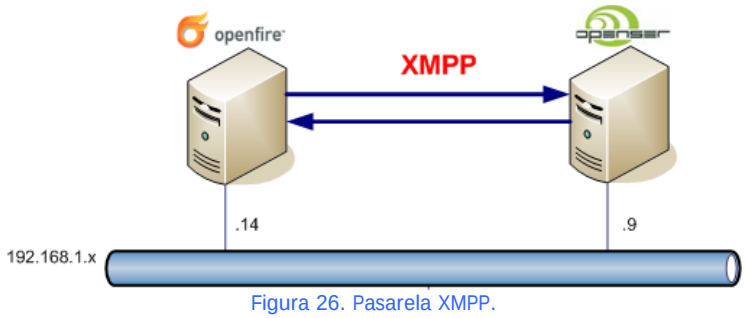


Figura 26. Pasarela XMPP.

Para establecer la pasarela XMPP, el servidor de Openser se deberá registrar como un componente externo en el servidor de Openfire.
Para establecer este registro se debe trabajar sobre la plataforma VoIP de Openser y sobre sus módulos.

El módulo de Openser que permite establecer la pasarela XMPP entre Openser y un servidor Jabber es el módulo “XMPP Module”. La documentación técnica de fabricante sobre como se implementa este módulo se puede revisar en el siguiente repositorio:

http://www.kamailio.org/docs/modules/1.3.x/xmpp.html
En el documento técnico de instalación anexo, “DT.I.05. Configuración pasarela XMPP”, queda ampliamente descrita la configuración de la pasarela XMPP diseñada para nuestro entorno de desarrollo.

Este diseño es fácilmente extrapolable para la futura plataforma de comunicaciones unificadas de la UPC.

A continuación se describe el funcionamiento de la pasarela XMPP, desde que se inicia un mensaje en la plataforma Jabber hasta que se recibe el mensaje en la plataforma SIP y de forma idéntica cuando partimos de un mensaje iniciado en la plataforma SIP y recepcionado en la plataforma Jabber.

Mensaje desde Jabber a SIP:

Cuando un cliente Jabber envía un mensaje a un cliente de IM de la plataforma SIP, se sigue el siguiente flujo:

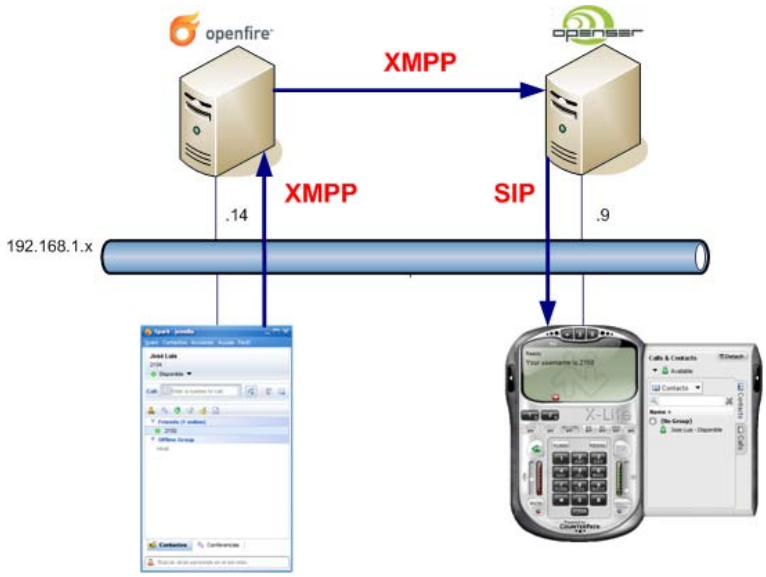


Figura 27. Tráfico mensaje Jabber-SIP.

1. Cliente Jabber envía mensaje a contacto de IM de la plataforma SIP:

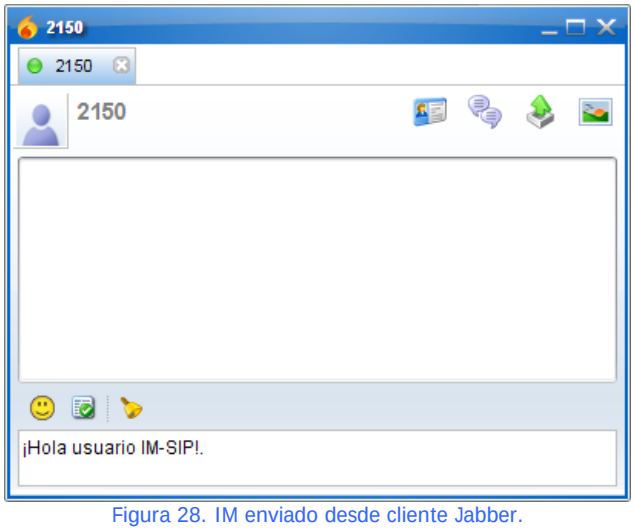


Figura 28. IM enviado desde cliente Jabber.

Al enviar el mensaje, el cliente jabber envía el siguiente paquete Jabber con el mensaje enviado:

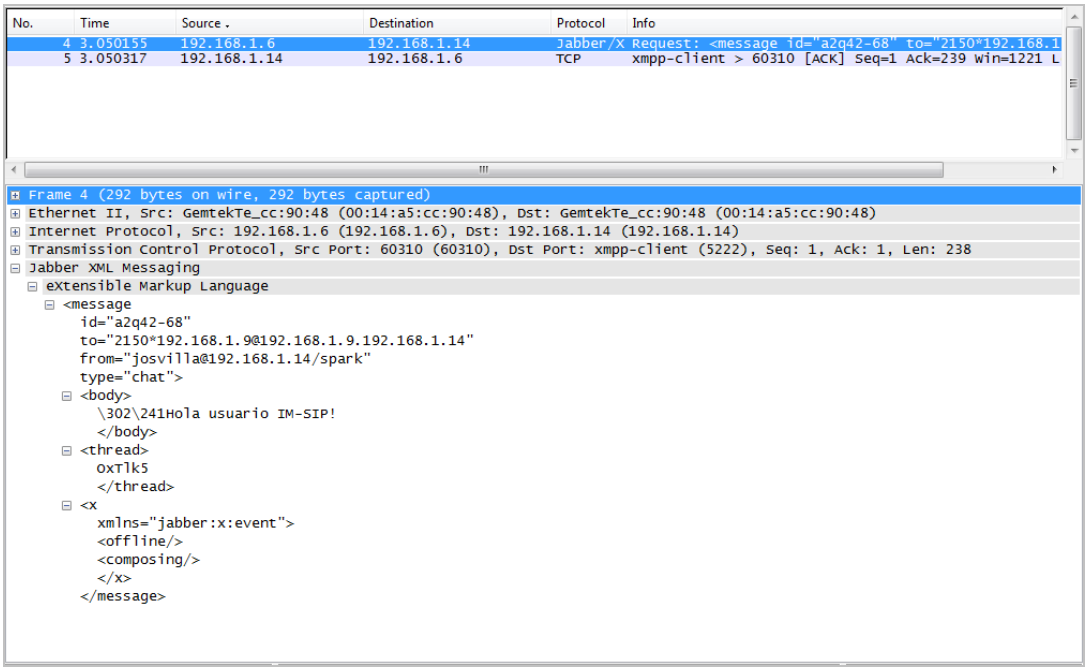


Figura 29. Paquete Jabber saliente con IM.

2. Openfire traspasa a Openser el paquete Jabber con el contenido del mensaje a través de la pasarela XMPP.

3. Openser entrega al cliente de IM el paquete SIP con el contenido del mensaje enviado por el contacto de la plataforma Jabber:

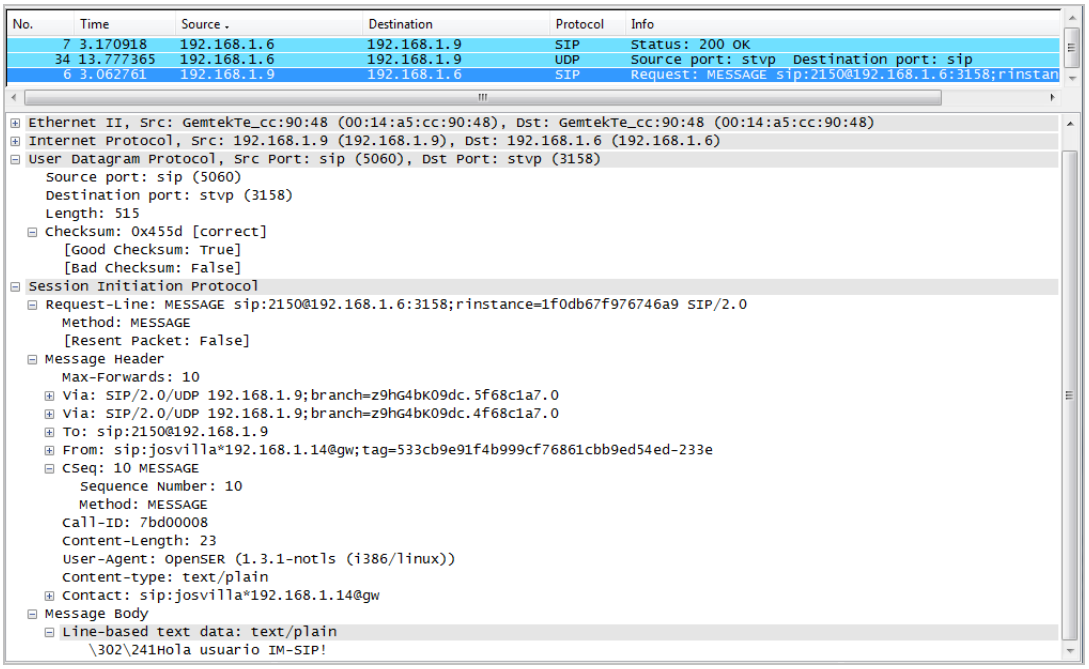


Figura 30. Paquete SIP entrante con IM.

Y junto con el paquete aparece un pop-up en el cliente SIP de IM con el contenido del mensaje:

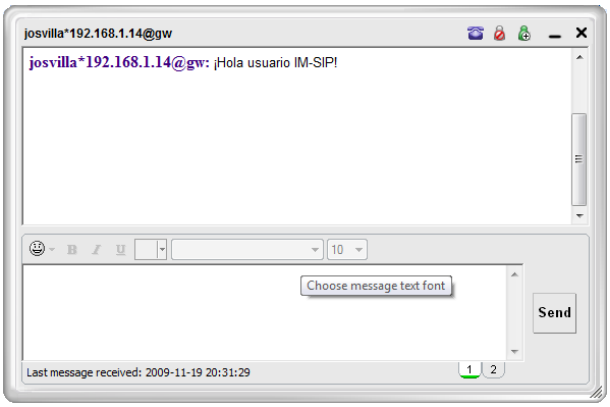


Figura 31. IM recibido en cliente SIP.

La transmisión de este mensaje XMPP-to-SIP, sigue el siguiente diagrama:

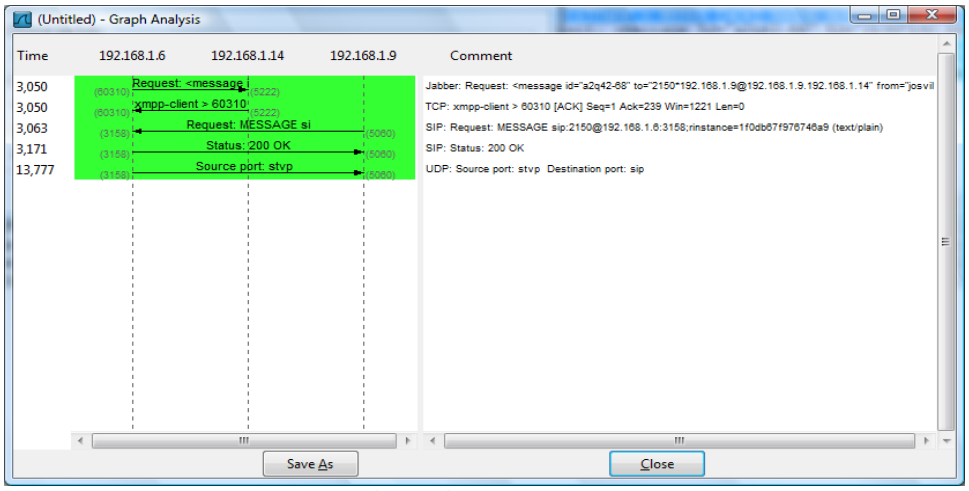


Figura 32. Análisis gráfico transferencia XMPP-SIP.

Mensaje desde SIP a Jabber:

Cuando un cliente SIP envía un mensaje a un cliente de IM de la plataforma Jabber, se sigue el siguiente flujo:

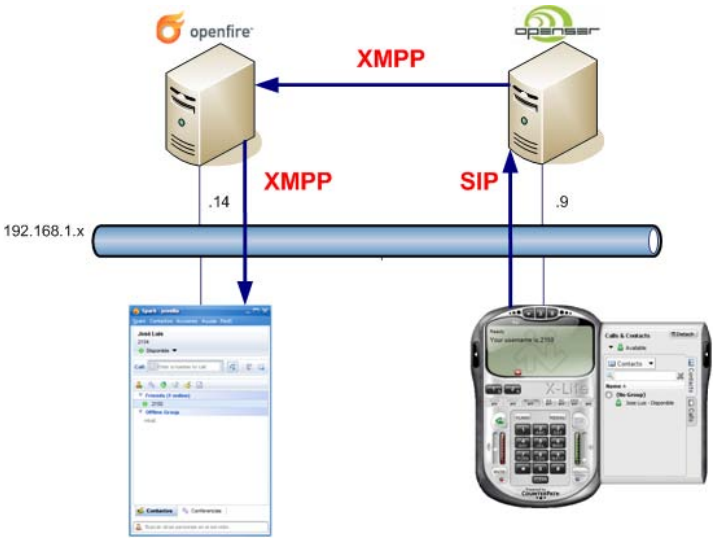


Figura 33. Tráfico mensaje SIP- Jabber.

1. Cliente SIP envía mensaje a contacto de IM de la plataforma Jabber:

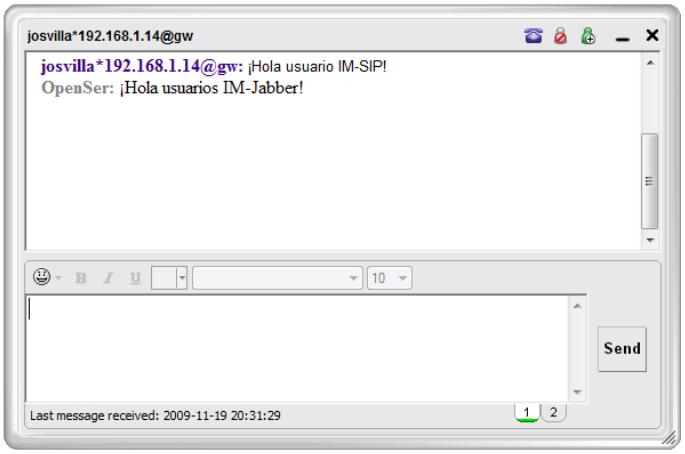


Figura 34. IM enviado desde cliente SIP.

Al enviar el mensaje, el cliente de IM-SIP envía el siguiente paquete SIP con el mensaje:

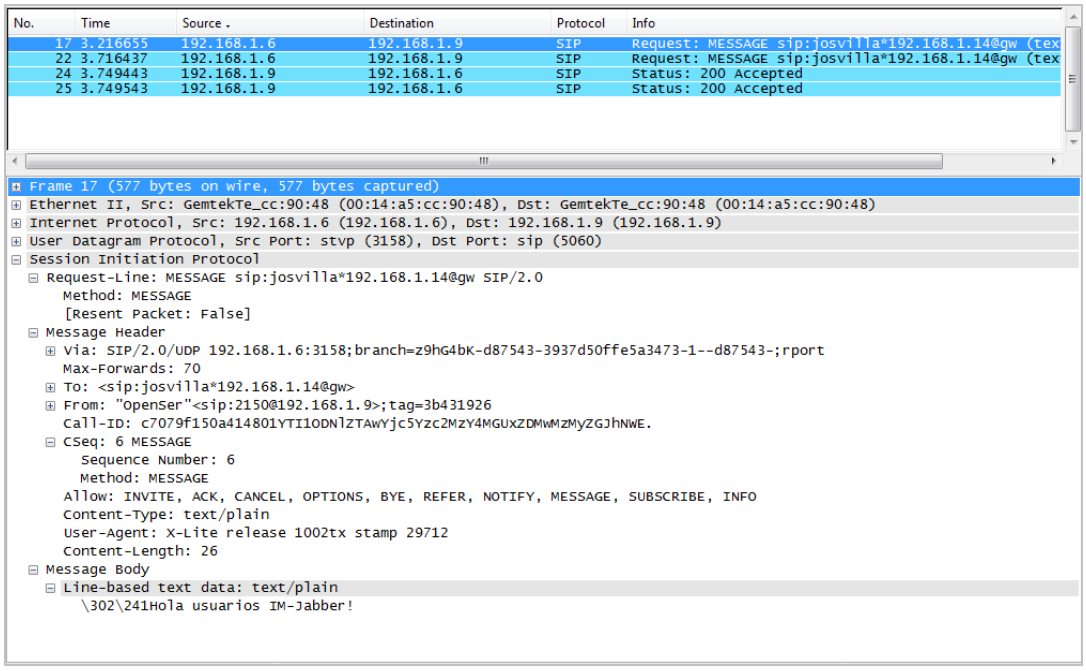


Figura 35. Paquete SIP saliente con IM.

2. Openser traspasa a Openfire el paquete Jabber con el contenido del mensaje a través de la pasarela XMPP.

3. Openfire entrega al cliente de IM el paquete Jabber con el contenido del mensaje enviado por el contacto de la plataforma SIP.

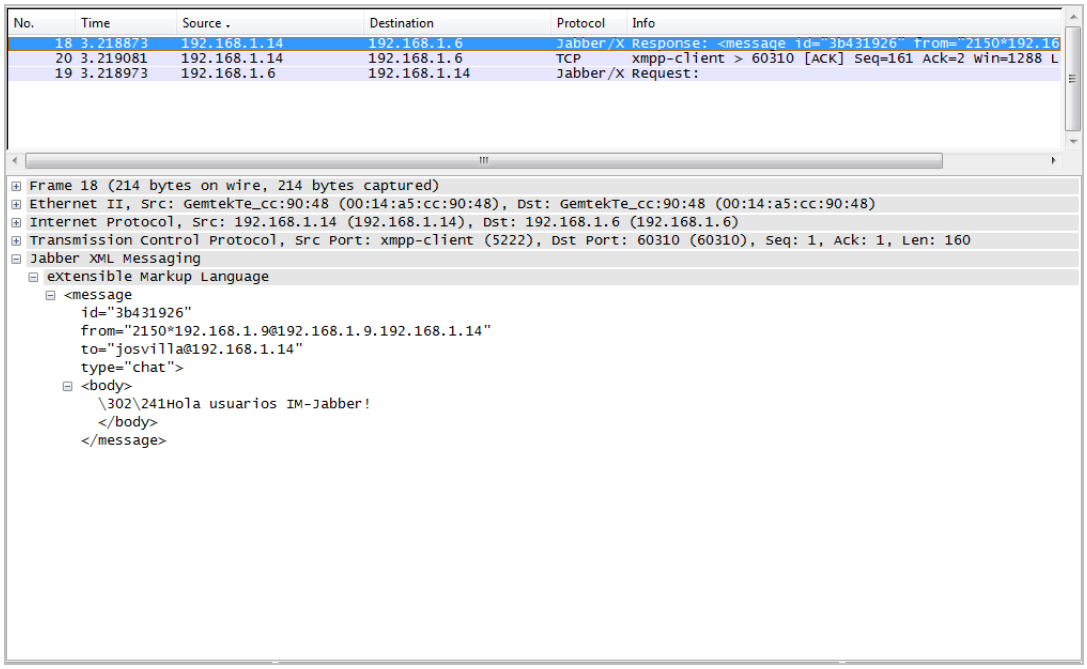


Figura 36. Paquete Jabber entrante con IM.

Y junto con el paquete aparece un pop-up en el cliente Jabber con el contenido del mensaje:

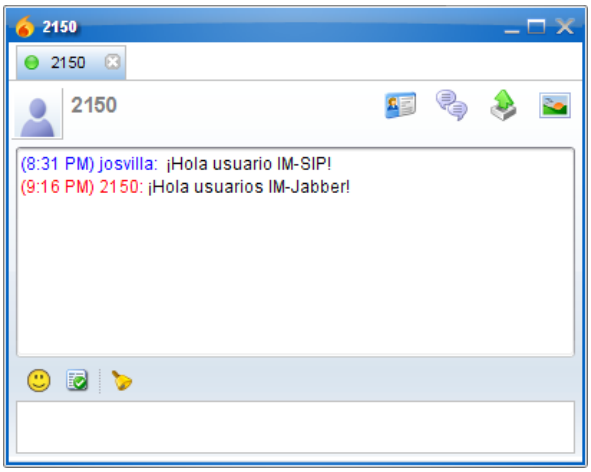


Figura 37. IM recibido en cliente Jabber.

La transmisión de este mensaje SIP-to-XMPP, sigue el siguiente diagrama:

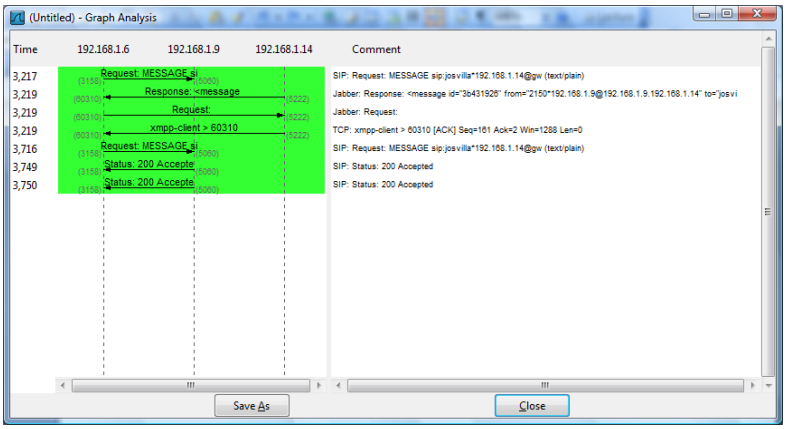


Figura 38. Análisis gráfico transferencia SIP-XMPP.

4.2.2 Traspaso de presencia entre las plataformas de VoIP e IM:

Para el traspaso de Presencia entre las plataformas de VoIP y de mensajería instantánea, se debe establecer una pasarela entre ambas plataformas.

Además de transmitir el estado de la presencia de los usuarios de IM, también se transmitirá el estado de los usuarios telefónicos, por ejemplo, si un cliente SIP esta atendiendo una llamada transmitirá su estado de presencia como que esta en estado “on the phone”, de esta forma si alguien quiere contactar con este usuario se esperará a que finalice su llamada, es decir cuando pasa a un estado IDLE.

Los estados básicos de Presencia que se implementan en ambas plataformas son los siguientes: IDLE o AVAILABLE, ON THE PHONE, DO NOT DISTURB, AWAY Y OFF LINE.

Para el traspaso de Presencia se hará uso de la misma pasarela XMPP que se usa para el intercambio de mensajería instantánea:

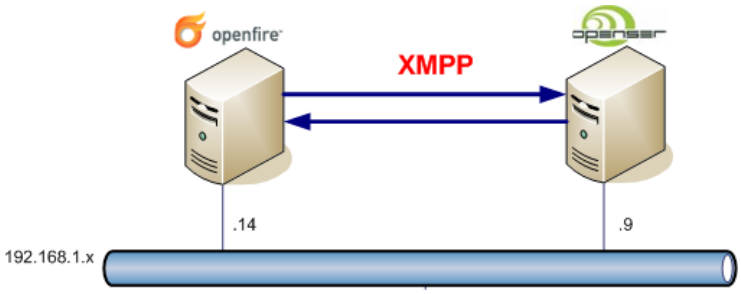


Figura 39. Pasarela XMPP para traspaso de presencia.

Para el traspaso de la presencia tal y como hemos dichos se hará uso de la pasarela XMPP, descrita en el punto anterior y explicada su implementación en el documento técnico de instalación anexo, “DT.I.06. Configuración traspaso de presencia”.
Además de usar esta pasarela, la plataforma de VoIP-Openser, necesitará un elemento que se encargue de la traducción de los mensajes XMPP a SIP y viceversa.

El módulo encargado de esta traducción es el módulo “PUA_XMPP Module”. La documentación técnica de fabricante sobre como se implementa este módulo se puede revisar en el siguiente repositorio:
http://www.kamailio.org/docs/modules/1.3.x/pua_xmpp.html

En el documento técnico de instalación anexo, “DT.I.06. Configuración traspaso de presencia”, queda ampliamente descrita la configuración de la pasarela para el traspaso de presencia diseñada para nuestro entorno de desarrollo.

Este diseño es fácilmente extrapolable para la futura plataforma de comunicaciones unificadas de la UPC.

A continuación se describe el funcionamiento del traspaso de presencia cuando se produce un cambio de estado en la plataforma Jabber, o como cuando se produce un cambio en la plataforma SIP.

Cambio del estado de presencia de un cliente Jabber:

Cuando un cliente Jabber cambia de estado pasando por ejemplo de Available a Do Not Disturb, la transmisión de mensajes siguen el siguiente flujo:

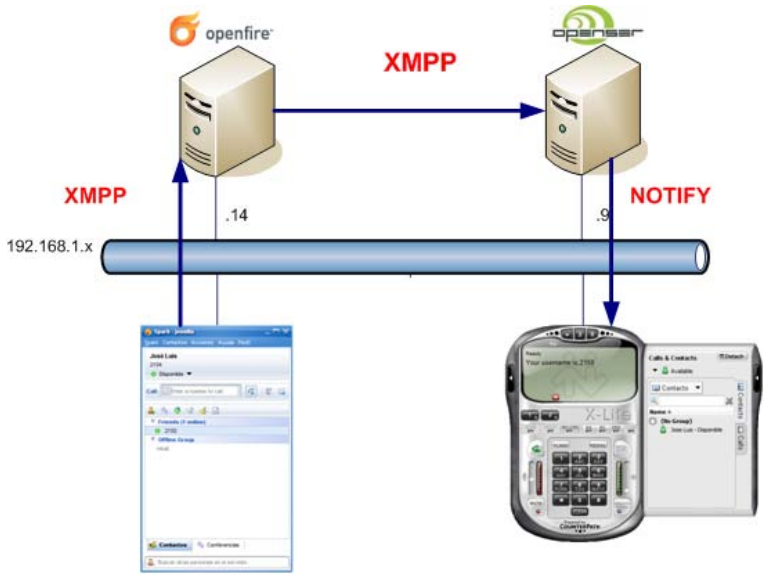


Figura 40. Flujo cambio de estado de presencia en cliente Jabber.

Este cambio de estado:

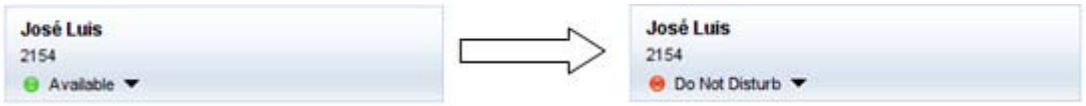


Figura 41. Cambio estado cliente Jabber.

Hace que se generen estos mensajes:

1. Desde el cliente Jabber al servidor Jabber:

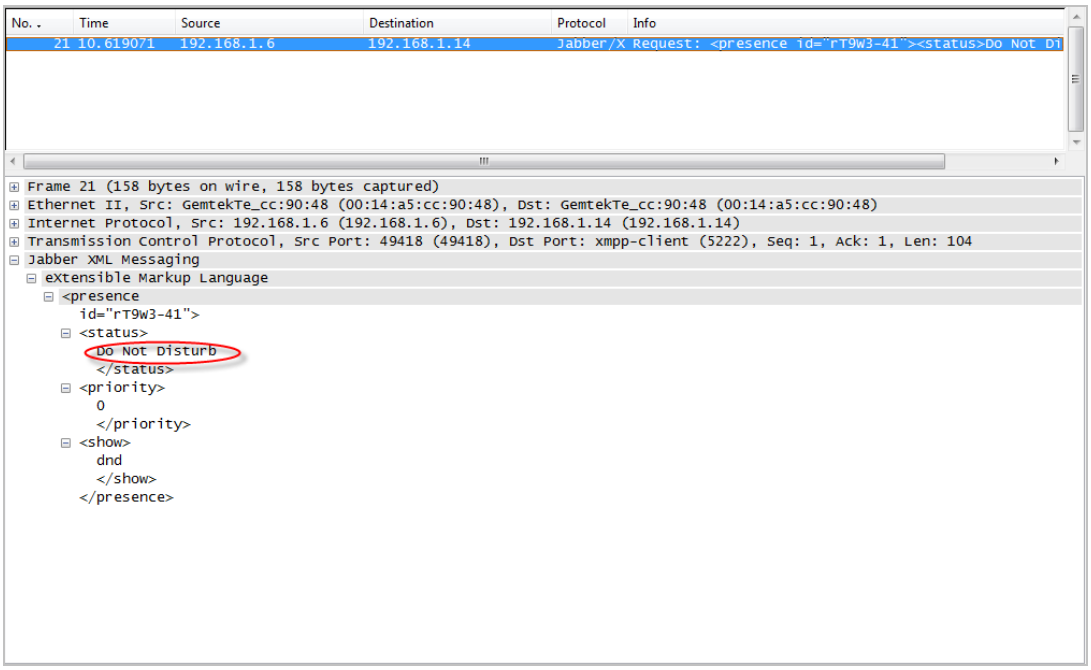


Figura 42. Traspaso cambio presencia cliente-servidor Jabber.

2. Desde el servidor SIP al cliente SIP:

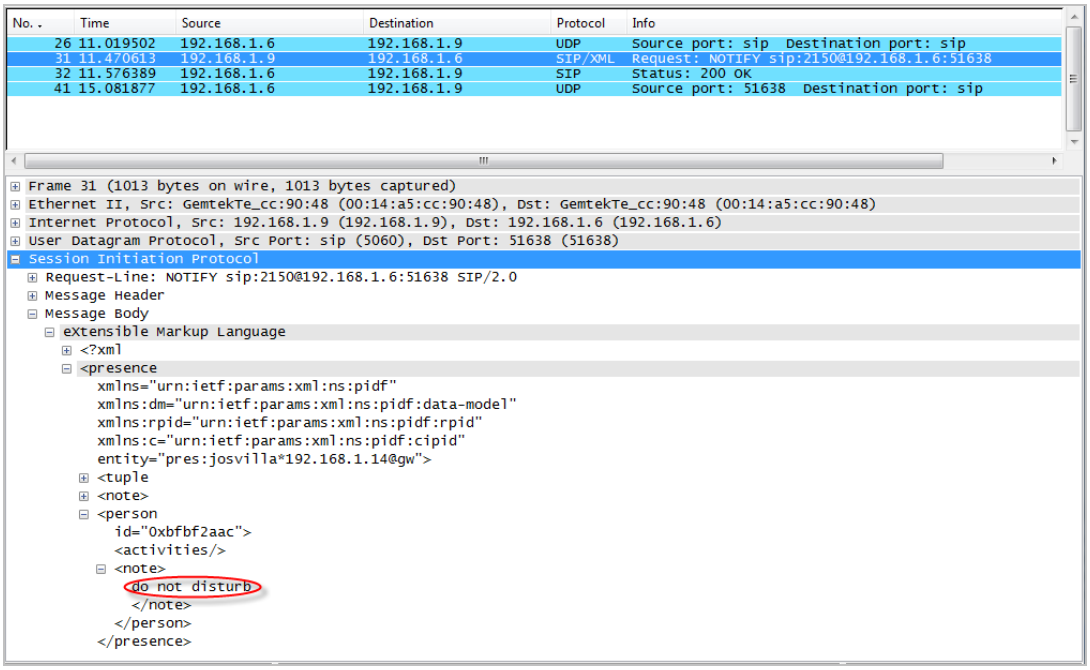


Figura 43. Traspaso cambio presencia servidor- cliente SIP.

El mismo mensaje de cambio de estado ha sido transmitido por la pasarela XMPP y convertido de XMPP a SIP, para finalmente ser notificado un cambio de estado en la Presencia de un contacto Jabber.

El traspaso de esta presencia sigue el siguiente diagrama:

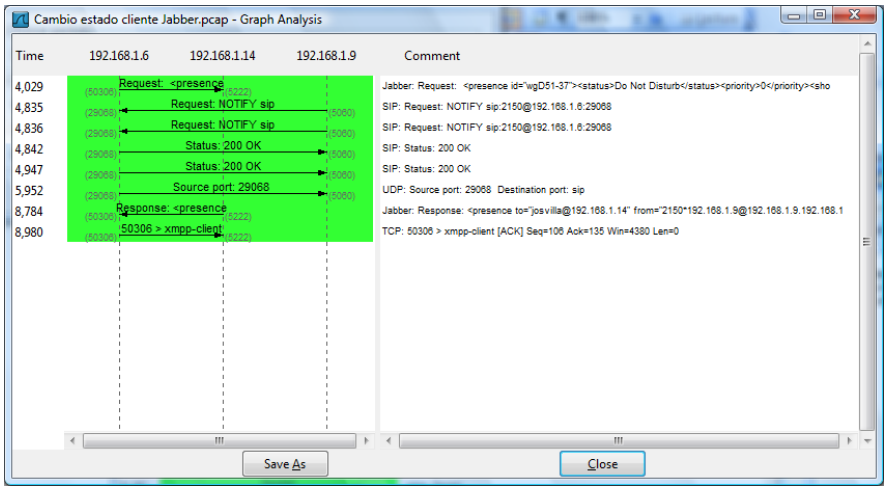


Figura 44. Análisis gráfico cambio de estado en cliente Jabber.

Cambio del estado de presencia de un cliente SIP:

Cuando un cliente SIP cambia de estado pasando por ejemplo de Available a On the Phone (cambio de estado típico para usuarios VoIP), la transmisión de mensajes siguen el siguiente flujo:

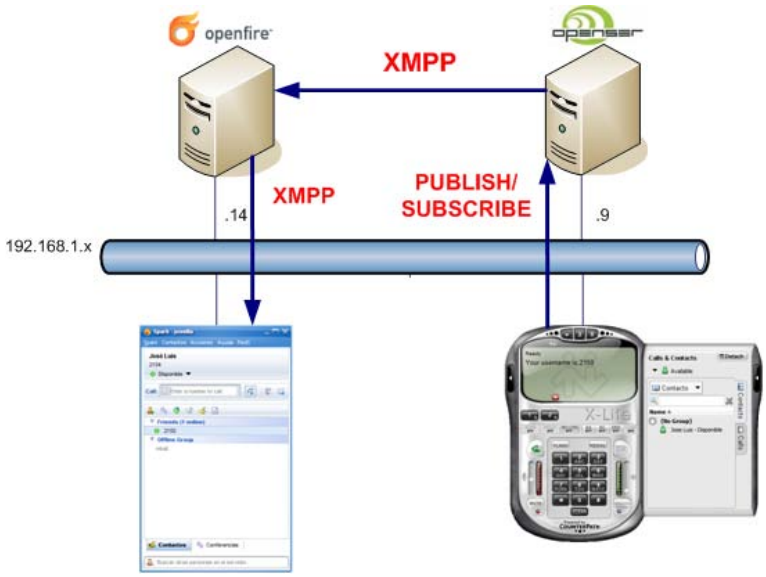


Figura 45. Flujo cambio de estado de presencia en cliente SIP.

Este cambio de estado:

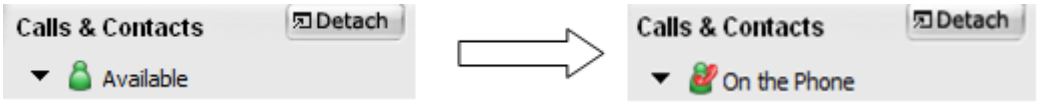


Figura 46. Cambio estado cliente Jabber.

Hace que se generen estos mensajes:

1. Desde el cliente SIP al servidor SIP:

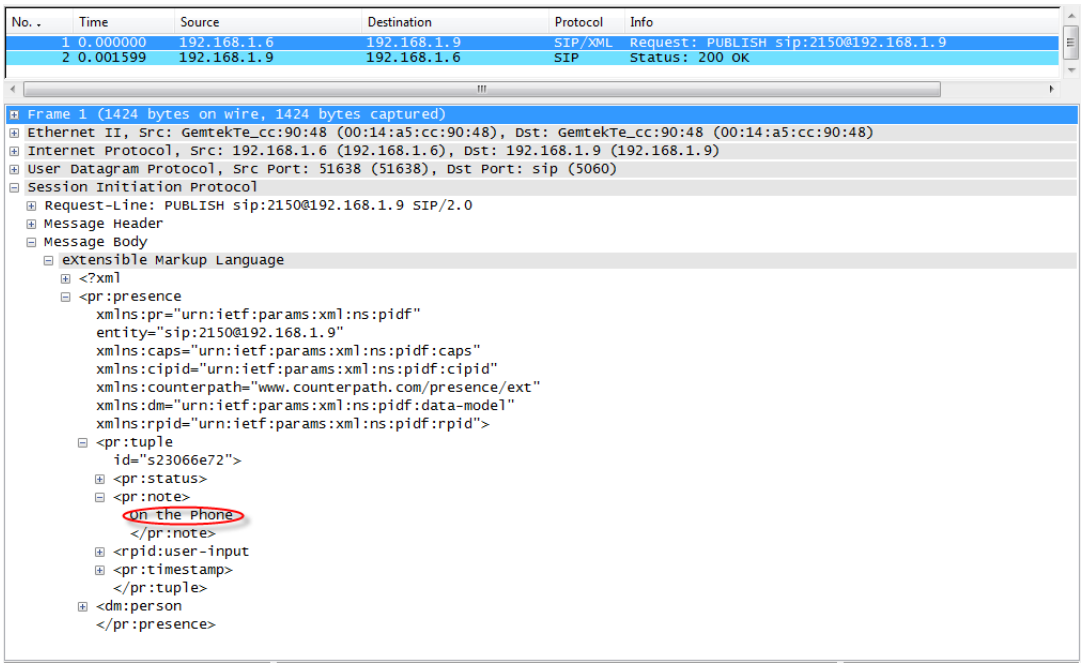


Figura 47. Traspaso cambio presencia cliente-servidor SIP.

2. Desde el servidor Jabber al cliente Jabber:

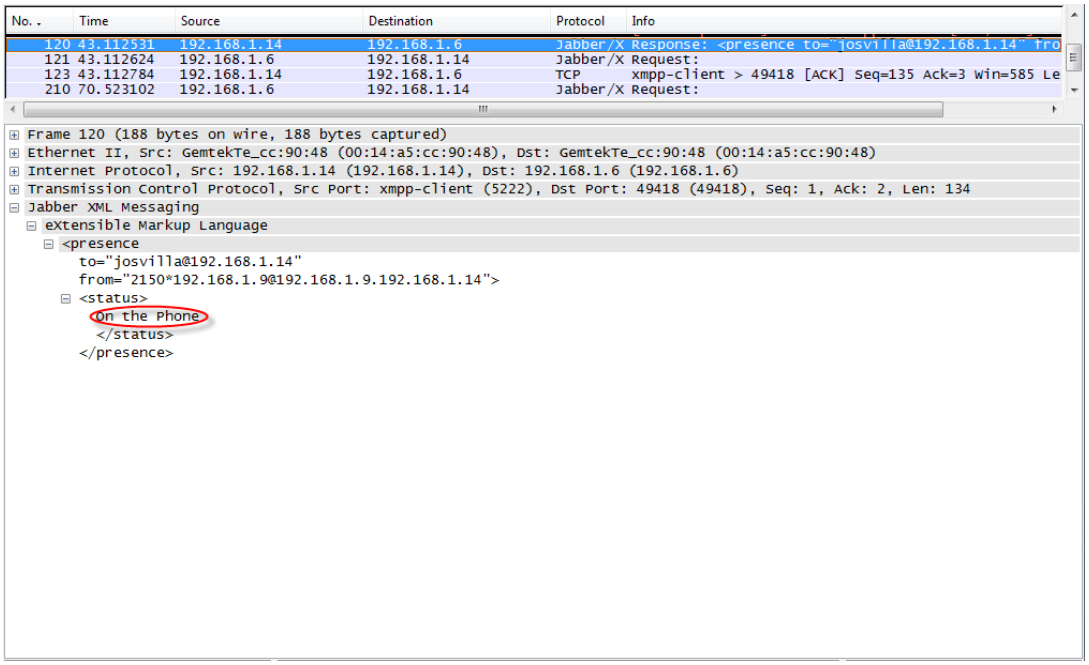


Figura 48. Traspaso cambio presencia servidor-cliente Jabber.

El mismo mensaje de cambio de estado ha sido transmitido por la pasarela XMPP y convertido de SIP a XMPP para finalmente ser notificado un cambio de estado en la Presencia de un contacto SIP.

El traspaso de esta presencia sigue el siguiente diagrama:

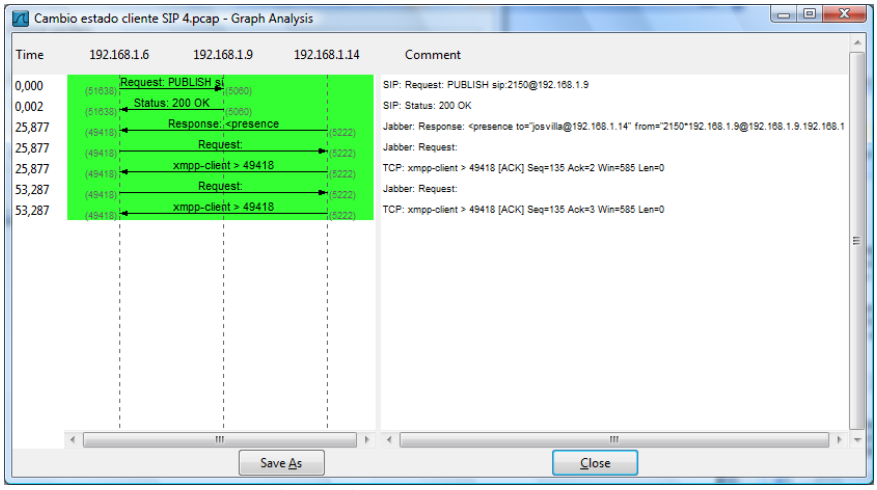


Figura 49. Análisis gráfico cambio de estado en cliente Jabber.

4.2.3 Gestión de usuarios a través de OpenLDAP:

El integrar la gestión de usuarios con una plataforma externa proporciona los siguientes beneficios:

- En fase de instalación no se han de dar de alta usuarios de forma manual. En grandes instalaciones es engorroso y poco productivo dedicar varias horas a dar de alta todos los usuarios de la plataforma.

- La gestión de usuarios queda centralizada en un solo punto.

- Los credenciales de acceso a la plataforma de mensajería instantánea serán los mismos que usarán los usuarios para registrarse en el dominio. Por lo tanto, tendremos una sola instancia de identificación. A esta característica también se le conoce como SSO (Single Sign On).

Openfire permite integrarse con varias plataformas de servicios de directorio, algunas de ellas son: Apache Directory Server, Apple's Open Directory, Fedora Directory Server, OpenLDAP y Microsoft Active Directory.

En nuestro entorno de trabajo virtual se ha trabajado la integración de la gestión de usuarios con OpenLDAP, ya que es esta la plataforma con la que UPCnet gestiona los usuarios de la UPC y por lo tanto era importante documentar como se hace esta integración.

Antes de configurar la integración en Openfire se ha de mantener una reunión con el administrador de sistemas que gestione el OpenLDAP de la UPC y solicitar los siguientes datos que vamos a necesitar a la hora de configurar Openfire:

- Host y Puerto: Necesitaremos conocer la dirección IP que tiene el servidor de OpenLDAP y el puerto sobre el que se comunicara con un third party.

- BaseDN: Es la ubicación de lo usuarios dentro del árbol organizativo de OpenLDAP.

- Usuario y Password: Para que Openfire consulte sobre OpenLDAP necesitará autenticarse con un usuario y password. Lo recomendable es que nos den de alta un usuario dedicado para la validación de Openfire.

En la fase de configuración de la integración con OpenLDAP, Openfire te permite testear que los credenciales para el registro en OpenLDAP son correctos:

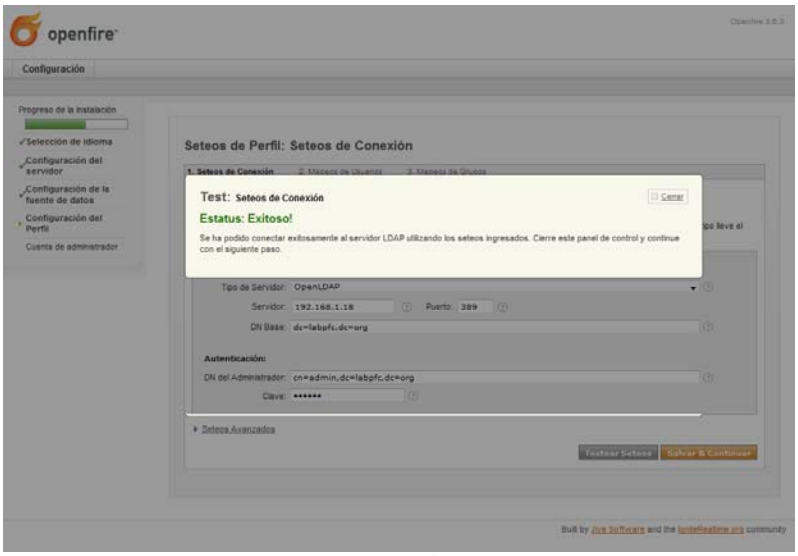


Figura 50. Testeo conexión con OpenLDAP.

Openfire te permite mapear la información de usuario que desees, es decir, no se tienen porque mapear todos los campos y quizás puede llegar a interesar mapear cierta información de forma particular:

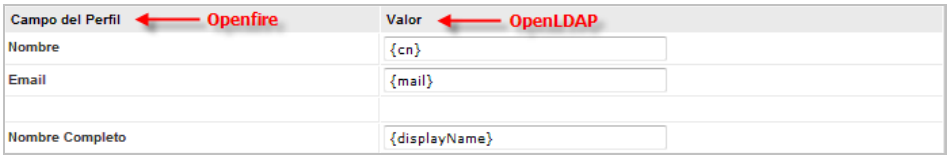


Figura 51. Mapeo campos OpenLDAP.

Tras el mapeo de usuarios, uno o varios de los usuarios mapeados deberán agregarse a la lista de administradores. ¡Muy Importante recordar los usuarios dados de alta como administrador!, sino no se podrá hacer login en la página web de administración de Openfire:

Figura 52. Mapeo usuario administrador.

En el documento técnico de instalación, “DT.I.08. Integración Openfire con OpenLdap”, queda descrito el procedimiento completo de integración de Openfire con OpenLDAP.

4.2.4 Integración usuarios SIP en clientes Jabber:

Openfire permite añadir funcionalidades al servidor de mensajería mediante la instalación de plugins. Openfire dispone de plugins gratuitos y de código abierto (la mayoría) y otros que son comerciales.

Uno de los plugins que resultan interesantes, es la integración de forma embebida de un cliente de VoIP SIP dentro del propio cliente Spark (cliente Jabber de Openfire). Este plugin es el plugin “SIP” que se puede descargar desde la misma web de IgniteRealtime o desde el propio servidor de Openfire

El plugin SIP, se encarga de redireccionar todo el tráfico SIP hacia el servidor Proxy SIP corporativo. De esta forma, desde el mismo cliente de mensajería instantánea se puede integrar un cliente SIP y por lo tanto, bajo una misma aplicación dispondremos de tráfico de IM y tráfico de voz.

El funcionamiento de este plugin es el siguiente:

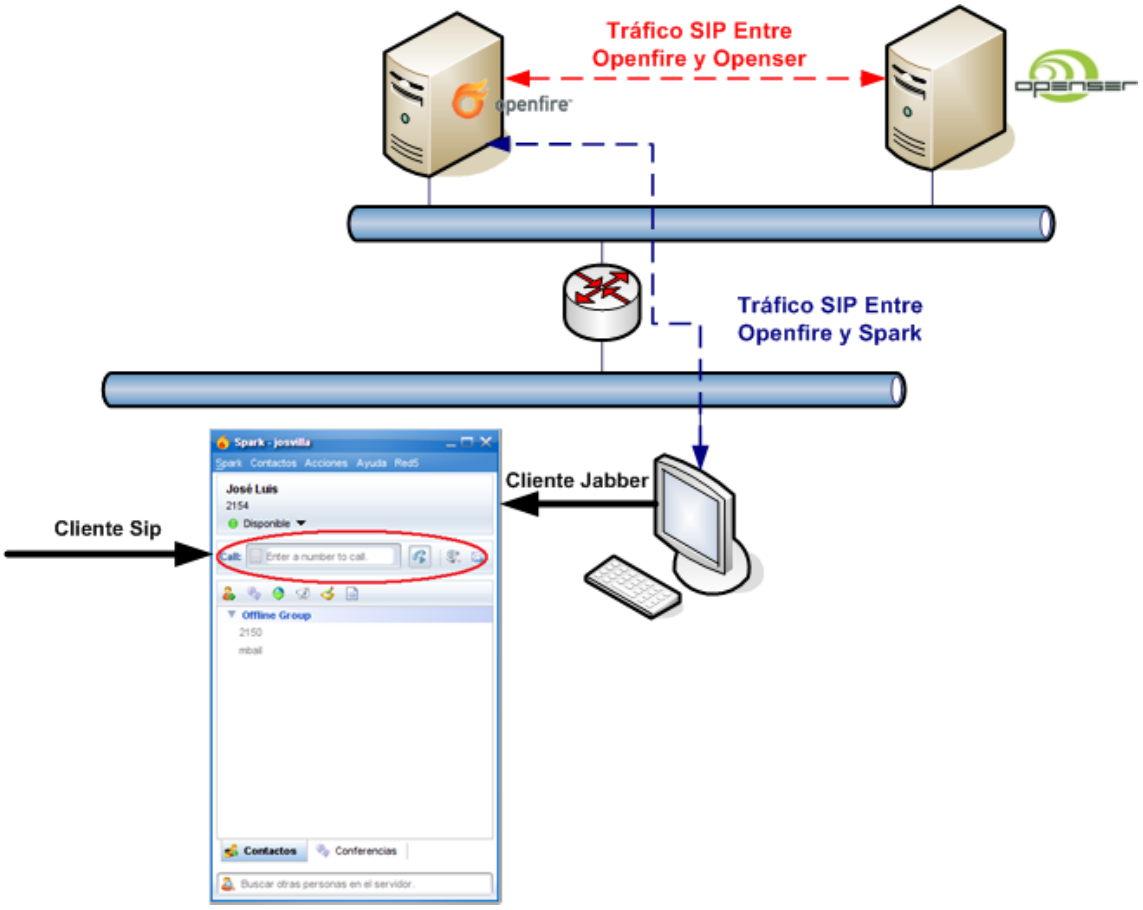


Figura 53. Esquema funcionamiento plugin SIP.

Hay dos flujos de tráfico SIP, uno entre Openfire y Openser y otro entre Openfire y Spark. Openfire se encarga de la asignación de usuarios SIP a usuarios Jabber y también se encarga de redireccionar todo el tráfico SIP que proviene de los clientes Spark hacia el servidor SIP.
Los clientes Spark, disponen de una interface donde tienen los controles de un softphone.

El aspecto visual del softphone embebido es el que se muestra en la siguiente página:

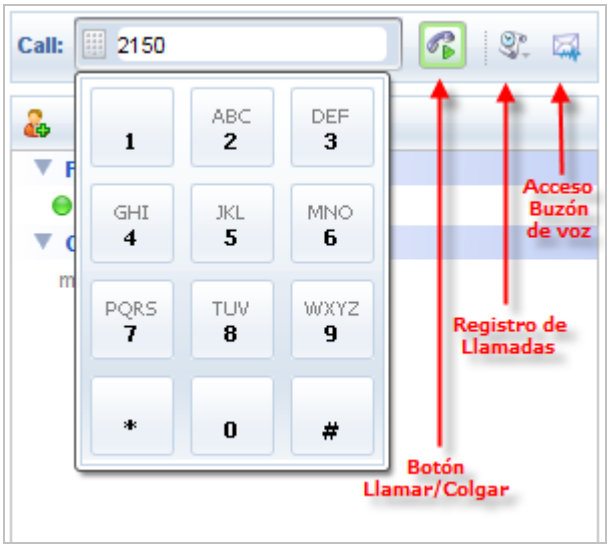


Figura 54. Softphone embebido en cliente Spark.

Se trata de un softphone muy sencillo y con unas funcionalidades básicas.

Una limitación de este plugin a tener en cuenta en la fase de diseño de la nueva plataforma de UC[15], es que no esta soportado la gestión de usuarios SIP en el caso que las plataformas de Openfire y Openser estén en distintos dominios. Por lo tanto, en el diseño de la plataforma de UC de la UPC, no se debe contemplar otra opción que no sea la de tener bajo el mismo dominio las dos plataformas.

Este plugin es gratuito y se puede descargar directamente de la web de IgniteRealtime:
http://www.igniterealtime.org/projects/openfire/plugins/sip.jar

La instalación del plugin se debe hacer tanto en el servidor, como en el cliente.

En el documento técnico de instalación, "DT.I.10. Instalación Plugin SIP", queda descrito el procedimiento de instalación.

4.2.5 Implementación videollamada desde Openfire:

Para la implementación de la videollamada desde Openfire recurriremos al plugin "Red5". Este es otro de los plugins gratuitos que se pueden descargar desde la misma web de IgniteRealtime.

El plugin Red5 de Openfire se basa en algunas funcionalidades desarrolladas para el proyecto Red5[16]. Red5 es un servidor Flash Open Source, encargado de entregar el contenido streaming en Flash. Para ello utiliza el protocolo RTMP (Real Time Messaging Protocol), con lo que se puede transmitir contenido en tiempo Real.

El protocolo RTMP es un protocolo propietario desarrollado por Macromedia y se usa para el streaming de audio, video y datos sobre Internet, entre un reproductor flash y un servidor.

Con el plugin Red5 se puede extender el soporte de Video y Audio a la funcionalidad de mensajería instantánea del servidor Openfire.

La instalación de este plugin queda descrita en el documento técnico de instalación, "DT.I.11. Instalación Plugin Red5".

Para la puesta en marcha de Red5 se ha de instalar un plugin en el servidor y otro plugin en el cliente Spark.
El funcionamiento de este plugin es el siguiente:

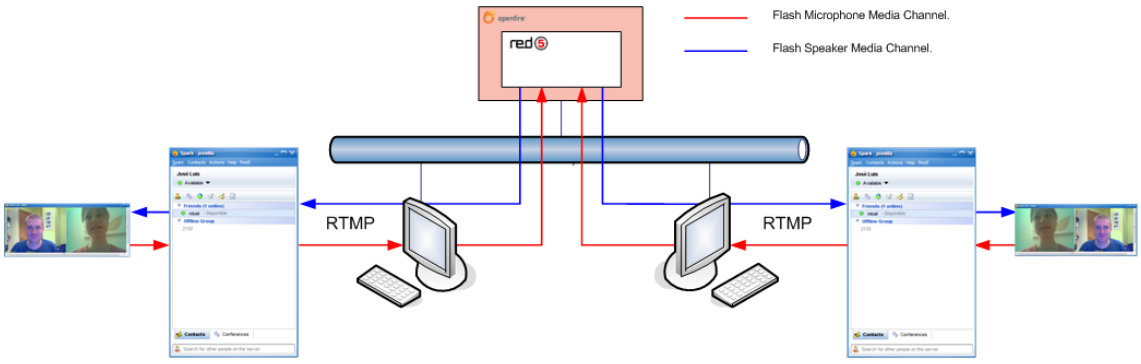


Figura 55. Esquema funcionamiento plugin Red5.

[15] UC, acrónimo Ingles de Unified Communications o en Castellano CU, Comunicaciones unificadas.
[16] RED5, http://www.osflash.org/red5

Al instalar el plugin Red5 en el servidor de Openfire, lo que se esta haciendo es instalar de forma embebida el servidor Flash Open Source de Red5. El streaming de video y audio entre el servidor y los clientes se hará a mediante el uso del protocolo RTMP.
La comunicación será full/duplex y habrá dos flujos simultáneos de tráfico RTMP, un flujo establecerá el canal de comunicación llamado “Flash Microphone Media Channel” y el otro flujo establecerá el canal de comunicación llamado “Flash Speaker Media Channel”.

El aspecto visual de las funcionalidades incorporadas con el plugin Red5 es el siguiente:

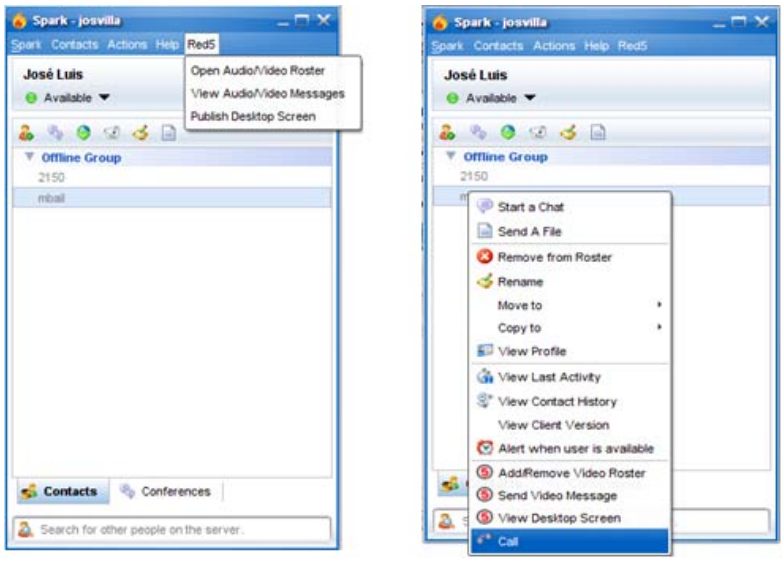


Figura 56. Funcionalidades Red5 en cliente Spark.

En el documento técnico de usuario, “DT.U.01 Funcionamiento cliente Spark”, queda descrito el funcionamiento del cliente de mensajería instantánea, incluido con el plugin Red5.

Además de la implementación de videollamada, este plugin tiene otras funcionalidades de valor añadido, tales como la publicación del escritorio y el envío de mensajes video.
Para publicar el escritorio es necesario tener instalada la máquina virtual de Java. A través de esta funcionalidad un usuario Spark puede mostrar a otro cliente Spark su escritorio. Esta funcionalidad puede ser muy útil es aspectos de soporte, tanto TIC como del propio campo al que pertenezca el usuario Spark.
El aspecto de un escritorio compartido a través de Spark es el siguiente.

Figura 57. Escritorio compartido con Red5.

Como se ha comentado antes, este plugin implementa la posibilidad de enviar video mensajes a un usuario, aunque no este conectado, de tal forma que cuando un usuario se conecte podrá reproducir todos los video mensajes que haya recibido, de la misma forma que si escuchara los mensajes del contestador de su teléfono.

4.2.6 Implementación cliente Sparkweb:

La disponibilidad en el repositorio de software de un cliente web para conectarse al servidor de mensajería instantánea, flexibiliza mucho la plataforma de mensajería instantánea, de tal forma que cualquier usuario podrá tener acceso con su cliente y su lista de contactos desde cualquier PC, sin necesidad de tener instalado el cliente Fat de Openfire.

El cliente web de Spark se conoce como Sparkweb, este cliente es gratuito y se puede descargar de la misma página web del fabricante, www.igniterealtime.com. Este software se puede instalar en cualquier servidor web, pero lo más coherente es aprovechar el Apache que tiene la plataforma donde esta instalado el propio Openfire.
Este cliente web tiene las funcionalidades básicas, y no implementa la videollamada.

El acceso al cliente de IM SparkWeb se podrá hacer desde cualquier plataforma, con cualquier sistema operativo y desde cualquier navegador web, accediendo a la dirección:
http://hostname/sparkweb/SparkWeb.html

En el documento técnico de instalación, “DT.I.03 Instalación cliente SparkWeb”, están descritos los pasos para la instalación de este software en un servidor web basado en Apache2 y sobre un Ubuntu Server 8.10, dado que se trata de la plataforma sobre la que se ha trabajado y es donde se ha hecho la instalación de la maqueta de la plataforma de Openfire.

En el documento técnico de usuario, “DT.U.02 Funcionamiento cliente Web de Mensajería Instantánea”, está descrito el funcionamiento del cliente web de spark a modo de manual de usuario.

El aspecto visual del cliente web es el siguiente:

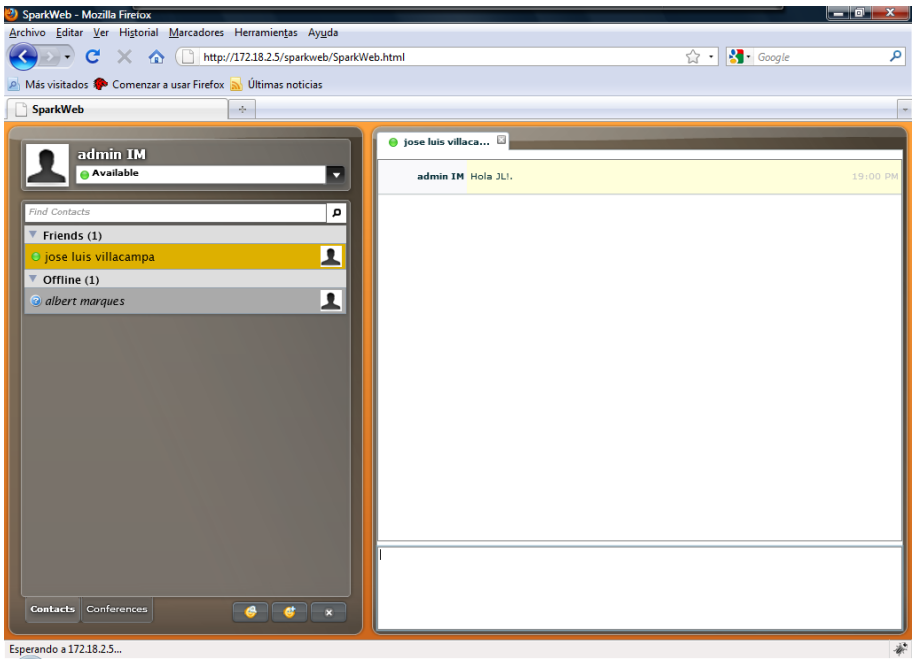


Figura 58. Aspecto visual cliente Sparkweb.

4.2.7 Alta disponibilidad y alto rendimiento:

La alta disponibilidad permite que un servicio funcione correctamente ante un fallo software o hardware. De esta forma, aunque se produzca un fallo en el sistema, el servicio seguirá funcionando de forma transparente para los usuarios como si el fallo no hubiera ocurrido. El fallo de un servicio que no dispone de alta disponibilidad provocará una parada del mismo, teniendo consecuencias muy negativas para el colectivo de usuarios de IM de la UPC.

La alta disponibilidad se puede implementar mediante una configuración hardware o software.
Una solución hardware trata de asegurar que el servicio funcione de forma interrumpida y para ello suelen utilizar sistemas redundantes de alimentación, discos duros (RAID), tarjetas de red, etc. De esta forma, si falla cualquiera de esos elementos hardware, el sistema funcionaría correctamente y lo único que tendremos que hacer es reemplazar el dispositivo averiado en "caliente" es decir, sin ningún tipo de parada del servidor que afecte al servicio.
Una solución software consiste en una serie de servidores, denominados nodos, conectados entre si de tal manera que, ante un fallo hardware o software, el servicio de IM ofrecido por uno de los nodos, es retomado por otro de los nodos del cluster. De esta manera el servicio que se ofrece sigue en funcionamiento de manera casi ininterrumpida.

En la siguiente figura se muestra un cluster de alta disponibilidad basado en software, formado por dos nodos dentro de un mismo segmento de red:

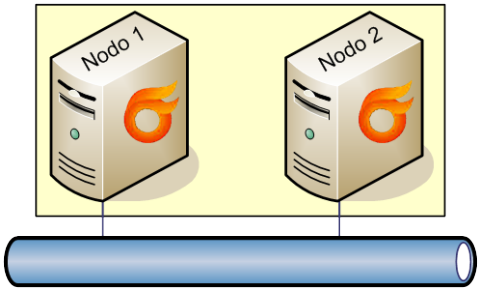


Figura 59. Cluster Openfire.

El alto rendimiento permite distribuir la carga de un servicio entre varios equipos, consiguiendo así un mayor rendimiento e impidiendo que se sature un servidor. Para permitir el alto rendimiento se crea un clúster que esta compuesto por nodos. Cada nodo se encarga de realizar una serie de tareas que se van distribuyendo dependiendo de diversos factores: equipos activos, rendimiento, etc.
Esta solución ofrece dos características muy importantes: La primera de ellas es que evita la saturación de una máquina. En un entorno corporativo como el de la UPC, una máquina que ofrece servicio a todos los campus, en momentos puntuales del día podría llegar a saturarse, debido al acceso masivo de los usuarios. Si la saturación alcanza picos muy elevados, la máquina podría fallar y el servicio pararse de manera inesperada. Una solución de alto rendimiento permitiría la posibilidad de repartir la carga entre varios servidores. A esta solución se le conoce como balanceo de carga.

Particularizando para el caso de Openfire, está disponible el plugin "Clustering". Este plugin permite crear nodos interconectados entre si, capaces de asumir la carga total en caso de fallo de uno de los nodos.

Para la instalación de este plugin es necesaria la instalación previa del paquete software: Oracle Coherence for Java. Este software se instala en el propio servidor de Openfire y concretamente dentro de sus directorios de instalación.

Además, si queremos disponer de una plataforma con alto rendimiento, podemos configurar el servidor DNS de la organización para balancear la carga usando la técnica del Round Robin. Esta técnica funciona respondiendo a la consulta DNS con una dirección IP distinta cada vez, con una dirección IP de cada uno de los nodos.

De todas formas esta técnica tiene dos desventajas: una de ellas es que no calcula la carga de cada nodo antes de redirigir el tráfico y la otra es que si uno de los nodos falla, su dirección sigue estando la lista de direcciones a resolver por el servidor DNS.
En GN/Linux existe una gran variedad de herramientas que proporcionan alta disponibilidad y resuelven el problema comentado pero, actualmente están proliferando las herramientas de virtualización y ya queda resuelto el problema de la alta disponibilidad.

En el documento técnico de instalación anexo, "DT.I.12 Instalación plugin Clustering", queda descrita la instalación de este plugin.

Desde la consola de administración de Openfire se habilita el clustering nodo por nodo. A continuación se muestra como queda configurado y monitorizado desde la misma consola, un clustering formado por dos nodos:

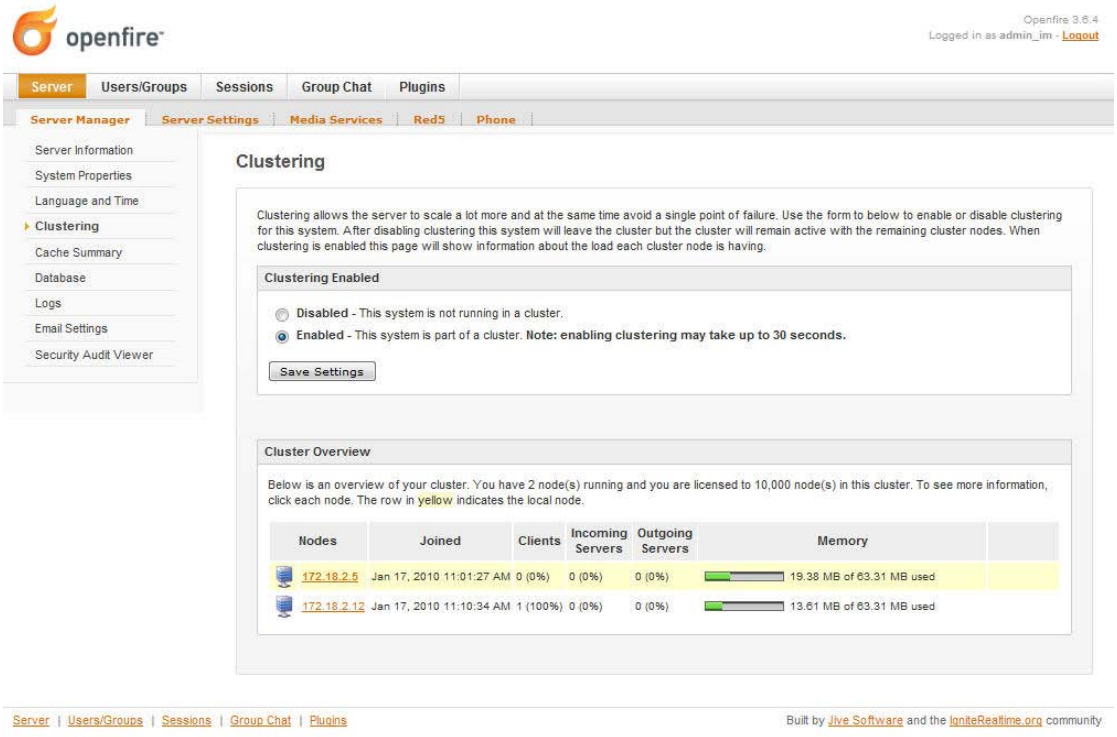


Figura 60. Gestión clustering.

4.2.8 Servicios de web collaboration para centro de contactos:

Un interface web permite establecer sesiones de atención al cliente vía web-chat, a esta forma de contactar con clientes o usuarios en tiempo real se le llama "web collaboration". Este entorno de atención web podría llegar a integrarse en un centro de atención de contactos con la aplicación encargada de gestionar el flujo de llamadas de las colas ACD[17].

Openfire, para proporcionar servicios de web collaboration dispone de los plugins Webchat y Fastpath. Ambos plugins forman el paquete Fastpath y por lo tanto la instalación de estos plugins se ha de hacer de forma conjunta, el instalar uno de ellos sin instalar los dos a la vez no proporciona ningún servicio añadido.
El plugin Webchat es el responsable de proporcionar el interface web de contacto y el plugin Fastpath es el encargado de gestionar el enrutamiento de las peticiones chat, también se encarga de gestionar las colas de agentes, en este caso los agentes son los clientes Spark.

[17] ACD, en entornos de contact center se define como cola ACD, la cola de distribución automática de llamadas.

Estos plugins inicialmente eran plugins comerciales pero se acabaron lanzando como Open Source. Ambos se pueden descargar de la propia web de IgniteRealtime y su instalación queda descrita en el documento técnico de instalación, “DT.I.13. Instalación plugins WebChat y Fastpath”.

Los componentes en un entorno de web collaboration basado en webchat y fastpath son los siguientes:

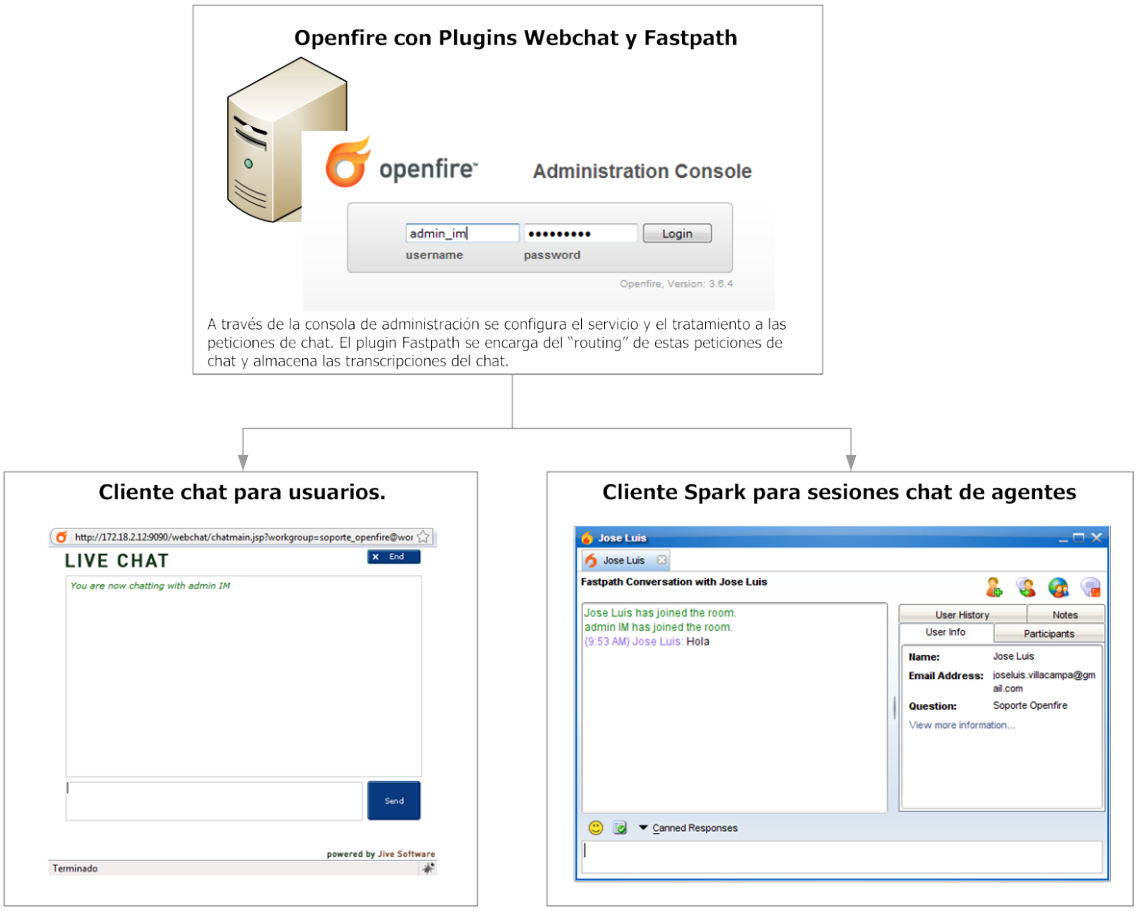


Figura 61. Componenentes entorno Web Collaboration.

El procesado de una petición de chat web sigue el siguiente flujo:

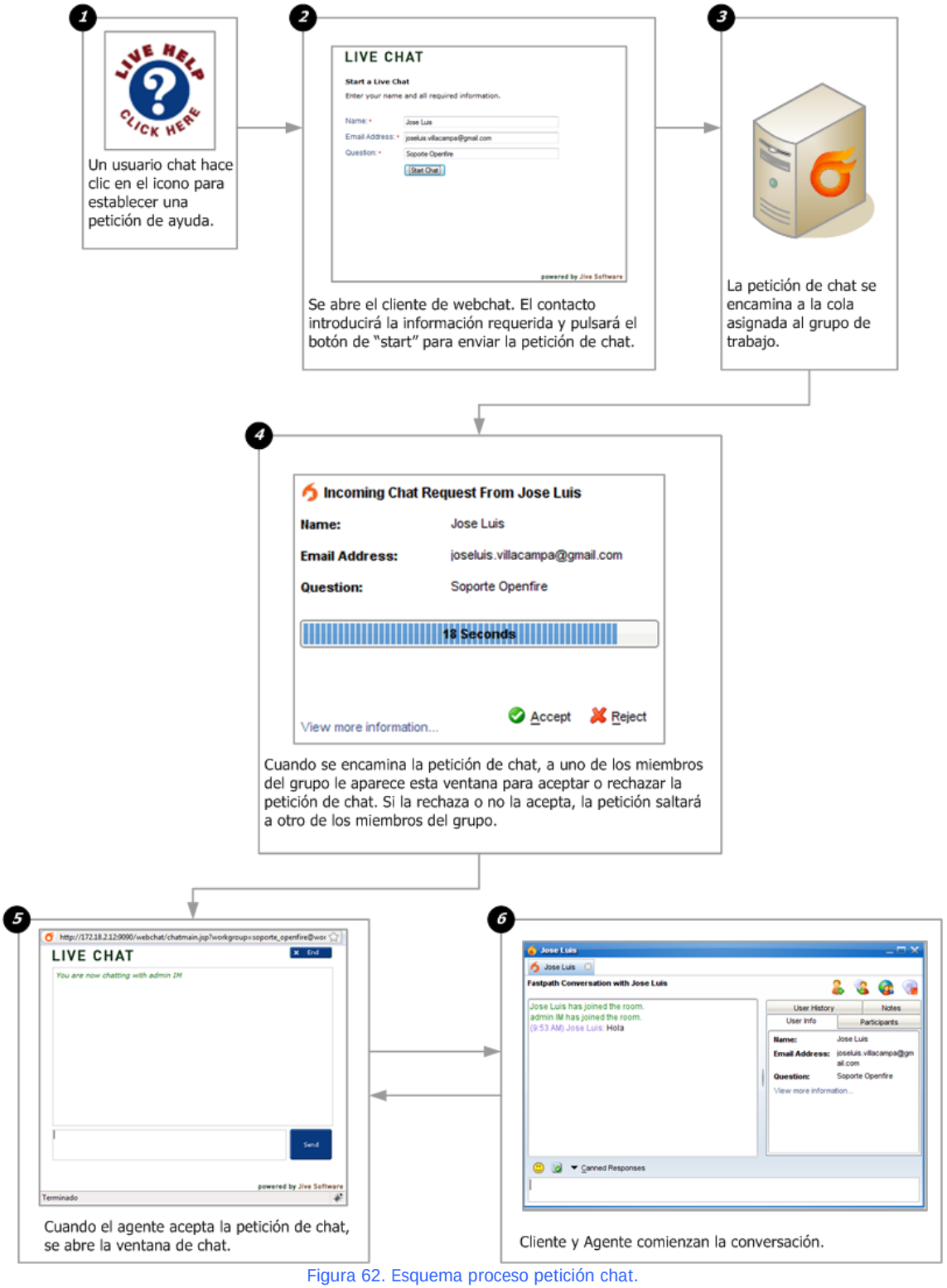


Figura 62. Esquema proceso petición chat.

El plugin Fastpath además de gestionar las peticiones de chat siguiendo determinados criterios, permite personalizar los iconos del chat y proporciona el código html para integrar los accesos dentro de la página web de la organización.

4.2.9 Escalabilidad de Openfire:

Openfire puede soportar hasta 5000 usuarios. Si las necesidades de una instalación requieren una plataforma con capacidad de gestión para más de estos usuarios, se deberá hacer un escalado de la misma para poder satisfacer las necesidades de la instalación.

Hay dos métodos de escalabilidad que se pueden utilizar con Openfire, uno de los métodos de que se puede utilizar es a través de “connection managers” o administradores de conexión y el otro método es a través de la formación de un cluster de servidores Openfire.

Un administrador de conexión es una plataforma externa al servidor Openfire que gestiona el registro de usuarios XMPP y se comunica con el servidor de Openfire. Esta plataforma puede gestionar hasta 5000 usuarios adicionales a los que Openfire puede gestionar por sí mismo. Sobre un mismo servidor Openfire se pueden conectar tantos administradores de conexión como se necesiten.

El diagrama de una instalación escalada a través de administradores de conexión sigue el siguiente esquema:

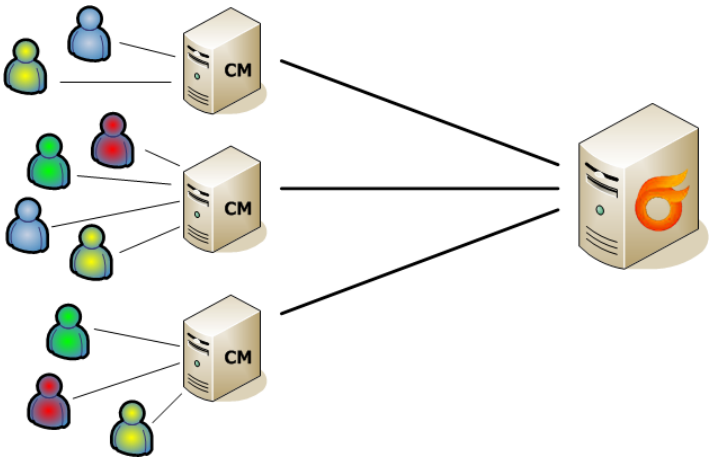


Figura 63. Esquema connection manager.

En la plataforma CM (connection manager) o administrador de conexión se instala el módulo connection manager de Openfire. Este módulo se puede descargar desde la misma página web de Igniterealtime, directamente desde este enlace:
http://www.igniterealtime.org/projects/openfire/connection_manager.jsp

Además de tener instalado y configurado un administrador de conexión, en el propio servidor de Openfire se debe habilitar la posibilidad de registro de administradores de conexión, por defecto esta deshabilitada, esto se hace bajo la pestaña de Server, Server Settings y Connections Managers:

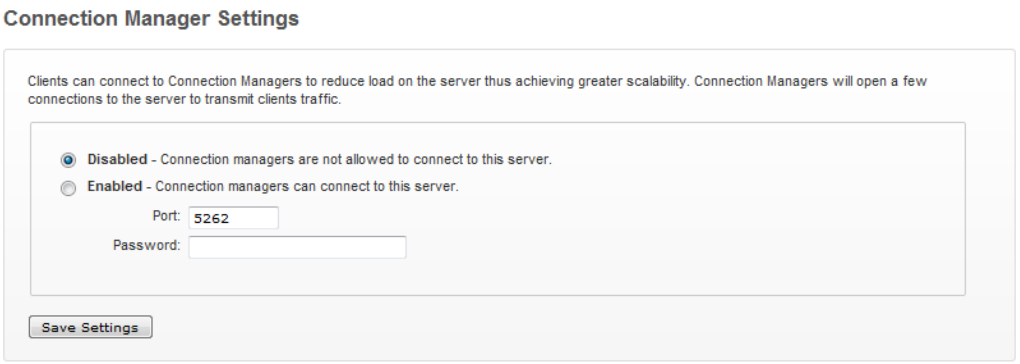


Figura 64. Parametros configuración Connection Manager.

El otro método para escalar una instalación es mediante el uso de clusters de servidores. El instalar un cluster de servidores de Openfire, además de aumentar el número de usuarios que pueden conectarse tiene otro beneficio más, el alto rendimiento. En caso de fallo de uno de los nodos el resto de nodos seguirán gestionando todo el tráfico XMPP, esto no ocurre con el uso de administradores de conexión.
Puesto que un servidor de Openfire soporta hasta 5000 usuarios, cada nodo de un cluster soportará hasta 5000 usuarios.

En sus inicios el plugin clustering no era Open Source y la herramienta Open Source para proporcionar escalabilidad en un sistema, era mediante el uso de los administradores de conexión.

En este trabajo no se ha testeado el uso de administradores de conexión ya que dado que el plugin clustering funciona muy bien, se ha testeado y los resultados son excelentes, no tiene sentido instalar administradores de conexión puesto que hay otra herramienta que además de escalar la plataforma proporciona alto rendimiento.
Por lo tanto, para una instalación global que proporcione servios en todos los campus UPC, se recomienda escalar el sistema a través de un cluster de servidores Openfire.

4.2.10 Pasarela con otras plataformas de mensajería:

Uno de los objetivos de que una organización disponga de su propia plataforma de mensajería instantánea, es precisamente el de bloquear el uso de sistemas de mensajería con acceso al exterior, el objetivo es evitar la perdida de tiempo que supondría el mal uso de un sistema de mensajería, sin perder los beneficios que supone disponer de este sistema.

Si bien es cierto a día de hoy la mensajería instantánea de Google, yahoo o Microsoft, esta plenamente extendida, por lo que a muchos usuarios el implantar un sistema de mensajería instantáneo sin acceso al exterior les puede suponer una barrera. Esta barrera en ciertos entornos de usuarios VIP, puede llegar a suponer la marcha a tras en la implantación de este tipo de plataformas. Por lo tanto, en este tipo de entornos puede ser interesante que estos usuarios no pierdan conectividad con el exterior y en cambio puede seguir interesando que al resto de usuarios les quede bloqueado el acceso. En estos casos, es necesario establecer una pasarela entre la plataforma Openfire de una organización con determinados servidores de mensajería instantánea y que permita limitarse el acceso a esta pasarela a unos pocos usuarios.

Para establecer esta pasarela, Openfire en sus inicios desarrollo el plugin “IM Gateway”. Este plugin permitía establecer pasarelas con unos pocos sistemas de mensajería instantánea, pero desde Octubre de 2009 ha dejado de tener soporte por parte de Openfire y ha sido sustituido por el plugin “Kraken”, bajo el proyecto Kraken. A pesar de este cambio de desarrolladores, el plugin sigue siendo Open Source.

El plugin Kraken esta escrito en Java y establece pasarelas XMPP con unos cuantos servidores más que su antecesor. Este plugin es la continuación del plugin IM Gateway.

Kraken ha sido testeado en este estudio y funciona perfectamente.

El plugin está disponible en el repositorio del sitio web del projecto Kraken, también están disponibles todas las versiones del plugin, pero la última versión (1.1.2) la podemos descargar desde este enlace:

http://sourceforge.net/projects/kraken-gateway/files/kraken-gateway/1.1.2/kraken.jar/download

Con el plugin instalado desde el administrador se puede habilitar el acceso a determinadas plataformas de mensajería instantánea y además, se puede habilitar el acceso a todos los usuarios, a uno o varios usuarios o a uno o varios grupos de usuarios.

También se puede configurar para que sea el administrador quien entre manualmente los credenciales de acceso a aquellos usuarios que lo tengan habilitado o que sean los propios usuarios los que introduzcan sus credenciales.

Una vez se valide un usuario en Openfire y haya introducido sus credenciales, en la barra del cliente spark aparecerán uno iconos nuevos representativos de las plataformas externas a las que están conectados. La siguiente imagen es un ejemplo representativo de un usuario que esta conectado a Google Talk y a Messenger:

Figura 65. Iconos plataformas IM externas.

Se podrá hacer “Sign out” de estas plataformas sin abandonar la sesión con Openfire.

Al establecer por primera vez sesión con una plataforma externa, se transferirá la lista de contactos de esta plataforma y si se accede vía cliente web Openfire, quedará establecida la sesión con la plataforma externa y también estará disponible la misma lista de contactos de la que se dispone desde el cliente Spark.

La instalación de este plugin queda descrita en el documento técnico de instalación, “DT.I.14. Instalación plugin Kraken”.

5. Implementación piloto de pruebas:

En el marco de este proyecto se ha implementado una plataforma piloto de pruebas. A través de este piloto se pretende recoger el feedback del funcionamiento e impacto de una plataforma de mensajería instantánea en el colectivo de usuarios de la UPC. Esta información servirá como experiencia a la hora de implementar la solución final que dé servicio a toda la UPC y ayudará a valorar que facilidades de valor que añadido pueden ser útiles de implementar y corregir las carencias que puedan surgir.

Para la puesta en marcha de esta plataforma piloto, debido a las buenas relaciones personales con el director de los Servicios Informáticos y las buenas relaciones y el marco de colaboración con UPCnet, se decidió poner en marcha esta plataforma en el Departamento de Ingeniería del Terreno, Cartográfica y Geofísica de la Universidad politécnica de Catalunya[18].

5.1 Entorno de la plataforma piloto:

El Departamento de Ingeniería del Terreno, Cartográfica y Geofísica, tiene su sede principal en el módulo D2 del Campus Nord de Barcelona. Su misión principal es desarrollar tareas de docencia e investigación, formando parte de la escuela de Caminos, Canales y Puertos de Barcelona[19] y colaboración con la Escuela de Arquitectura Técnica.[20]

Las áreas de investigación y docencia del departamento tratan básicamente la experimentación teórico-práctica del terreno, así como la definición de modelos matemáticos que describan y puedan predecir sus comportamientos. El uso de la informática, especialmente para el cálculo intensivo, es una herramienta crucial para el desarrollo de estas tareas.

Los Servicios Informáticos del departamento (SIETECG), se ubican en el módulo D2 del Campus Nord y se encargan de gestionar todos los recursos TIC (Tecnologías de la Información y Comunicación) para poder implementar servicios de calidad que den soporte a las tareas propias del departamento. Estos servicios se pueden enmarcar en los siguientes grupos:

- Servicios de acceso a red, donde se encuentra la administración de la red local y el acceso a la red troncal de la UPC, con salida al exterior.
- Servicios de ficheros e impresión, donde están los sistemas que permiten el trabajo colaborativo en grupo, así como el uso de recursos compartidos y los permisos adecuados.
- Servicio de cálculo intensivo, donde se gestiona todo lo relativo a las máquinas de cálculo intensivo, desde la decisión de la configuración, su compra, instalación, mantenimiento y reciclaje.
- Servicio de configuración y mantenimiento de estaciones de trabajo de usuarios, donde se gestiona toda la vida de uso de una estación personal, la compra, instalación, mantenimiento software y hardware, solución de problemas y retirada y posterior reúso o reciclaje.

[18] ETCG, www.etcg.upc.edu
[19] ETSECCPB, www.camins.upc.edu
[20] EPSEB, www.epseb.upc.es

5.2 Topología de red en ETCG:

La topología de red del Departamento sigue un diseño clásico de firewall de 4 subredes:

- Red Wan o acceso al exterior, se trata del enlace troncal con la UPC y corresponde con la red 147.83.51.0/24.
- Red LAN o red local, es en este segmento donde están todas las estaciones de trabajo de todos los usuarios del departamento a excepción de de las estaciones de trabajo de secretaría, que por razones funcionales estarán en otro segmento. En este segmento se utiliza un direccionamiento de clase B, concretamente el direccionamiento 172.16.0.0 /16 (hasta 65.534 equipos).
- Red DMZ o red desmilitarizada, es el segmento de red donde están instalados los servidores y servicios comunes (impresoras, faxes, fotocopiadoras, etc).
- Red SECR ETCG, es el segmento de red donde se alojan las estaciones de trabajo del equipo de secretarías. Estas máquinas disponen de direccionamiento público debido a que para usar el ERP (SAP R3) de la UPC se requiere de una dirección IP real de la UPC. En este segmento se utiliza un direccionamiento de clase B, concretamente el direccionamiento 147.83.174.52/29 (hasta 6 equipos).

Se consideran usuarios remotos a los usuarios de Campus Sud y a un grupo de usuarios que se encuentran ubicados en una oficina de la calle Josep Maria de Segarra. Estos usuarios remotos se conectan a la red del Departamento mediante accesos VPN.

El esquema topológico de la red del departamento es el siguiente:

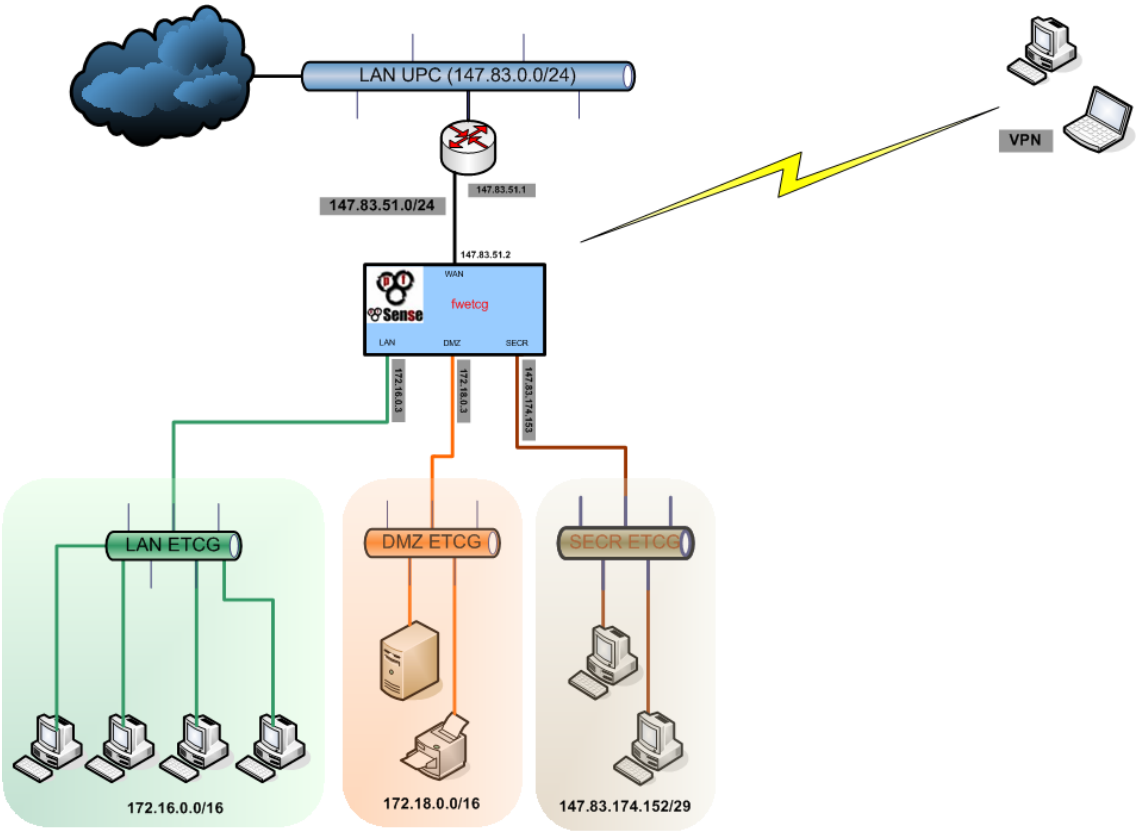


Figura 66. Esquema topología de red en ETCG.

5.3 Toma de requerimientos:

En el mes de septiembre se mantuvieron varias reuniones con el director de los servicios informáticos de ETCG. En estas reuniones se expusieron los servicios de valor añadido que se pueden implementar en la plataforma de Openfire y las topologías redundantes posibles que se pueden implementar.

Durante el seguimiento y confección del perfil de la nueva plataforma, se percibió el interés y necesidad que tenían en ETCG. Muchos de los usuarios viajan constantemente, trabajan desde casa a través del VPN o incluso en periodos vacacionales de forma excepcional entran en contacto con el entorno laboral. Muchas veces necesitan contactar con alguien de forma rápida y eficiente para hacer consultas en tiempo real. Para este tipo de usuarios, que pueden encontrarse fuera de España es muy importante de disponer de un elemento de comunicación corporativo, paralelo al uso del teléfono y de características similares, con la correspondiente reducción de gastos que suponen las llamadas y/o conferencias internacionales.

Tras finalizar la toma de requerimientos, se perfiló una plataforma que tuviera las siguientes características:

- Plataforma de mensajería instantánea robusta, con una administración fácil y amigable.
- Implementación sobre última versión Ubuntu Server LTS.
- Alta disponibilidad hardware.
- Permita el acceso remoto para usuarios y administradores.
- Integración con el directorio activo del departamento.
- Implemente videollamada.
- Acceso a la plataforma de mensajería a través de un cliente web.
- Posibilidad de acceso a plataformas IM externas, tales como MSN o Google Talk.

Inicialmente se descarta la implementación de un sistema en alta disponibilidad a través de plataformas en topología de cluster debido a tres motivos fundamentales, principalmente porque solo se dispone de una máquina para implementar este servicio, el servicio de mensajería instantánea corporativo no es fundamental para el desarrollo de las actividades del departamento y además, la máquina sobre la que se va a implementar esta plataforma es servidor con doble fuente de alimentación, discos en Raid 5 y doble tarjeta de red, con lo que ya queda implementada una redundancia hardware que ofrece robustez ante fallos hardware de la máquina.

Tras finalizar la puesta en marcha de la plataforma y previo a la puesta en producción se ha requerido la entrega de los siguientes manuales: Manual de usuario del cliente Spark y del cliente Sparkweb, y unos pequeños trípticos resumiendo el funcionamiento de ambos clientes, estos documentos se entregaran al director de los servicios informáticos y se distribuirá a sus colaboradores y al colectivo de usuarios de ETCG.

También se requiere de una formación a los administradores de los servicios informáticos sobre la administración de Openfire y sobre el funcionamiento del cliente Spark. En una de las reuniones se advierte que a la parte de la formación del cliente Spark, podrá acudir algún usuario VIP, dependiendo de la disponibilidad de este. Al resto de usuarios no está prevista ninguna formación ni por parte del proyectista ni por parte de los administradores de SIETCG. Se considera que con los manuales de funcionamiento y sus trípticos será suficiente.

5.4 Instalación de Openfire:

La instalación de Openfire en ETCG está dividida en tres pasos:

- Instalación de sistema operativo adecuado para instalar Openfire.
- Instalación de Openfire.
- Instalación servicios de valor añadido requeridos por ETCG.

Previo a la instalación del SO, se deberá preparar la máquina virtual sobre la que se instalará Openfire. Se trata de una instalación personalizada según los requerimientos de la dirección de SIETCG.

5.4.1 Instalación máquina virtual:

En ETCG para proporcionar los siguientes servicios:

Simba, controlador de dominio SIETCG
VPNEtcg, servidor de acceso remoto al departamento
WebMIVES, servidor web de www.mives.upc.es
VMWare Admin, estación de trabajo de acceso remoto para administrar el VMWare
WebLLISCAT, servidor web de www.lliscat.upc.es
WebPRH2O, servidor web de www.proyectosh2o.upc.es
Web2etcg, servidor web de www2.etcg.upc.es
Webhidrologia, servidor web de www.h2ogeo.upc.es
SubVGHS, servidor de Subversión para el grupo de hidrológica subterránea.

tienen instalado VMware ESXi 4.0 instalado sobre un servidor de la marca Dell con las siguientes características:

Dell PowerEdge

- Procesador doble Intel Xeon E5520.
- 32 Gb de memoria RAM.
- 8 discos de 1Tb SATA 7200 3,5" en congiruación de Raid 5.
- Controladora raid PERC 6/i con 256 MB de caché.
- Dos fuentes de alimentación redundantes de 870 W.
- Tarjeta de video Matrox G200 integrada (8 MB de vídeo).
- Adaptador de red Intel PRO/1000 PT, dos puertos, Gigabit, PCI-E x4.

Sobre este VMware se ha virtualizado una nueva máquina dedicada para la instalación de Openfire con las siguientes características:

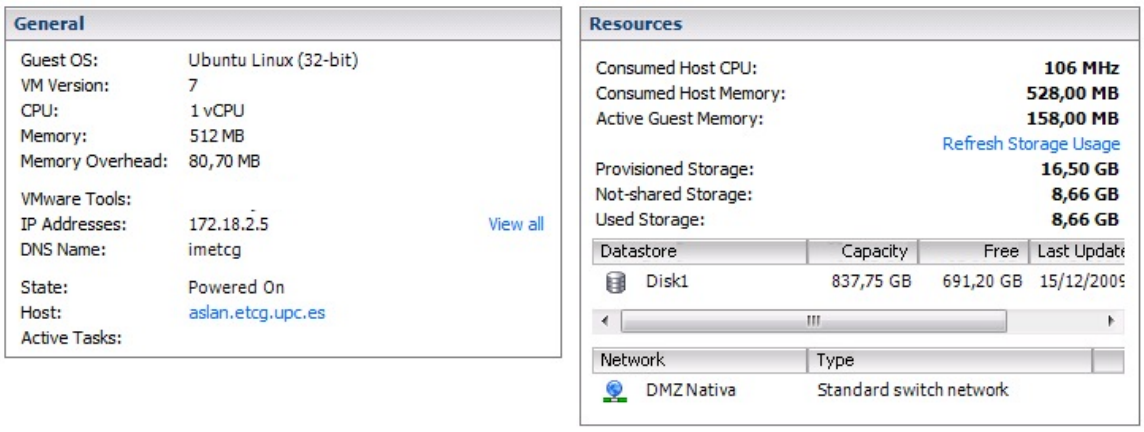


Figura 67. Características máquina virtual IMETCG.

Las propias características físicas del servidor, hacen que quede implementada una redundancia hardware. Además, el hecho que Openfire este implementado sobre una máquina virtual hace que de cierta forma quede implementada, de manera limitada, una redundancia software a través de la posibilidad de la toma de múltiples Snapshots.

5.4.2 Instalación servidor Linux:

La instalación de Openfire se ha hecho sobre una distribución Linux. Los requerimientos de SIETCG eran que se hiciera sobre la última distribución Ubuntu LTS disponible. A fecha de octubre de 2009, la última distribución Ubuntu LTS disponible es la versión 8.04.
El Roadmap de Canonical anuncia que la nueva distribución Ubuntu LTS, tanto para la versión Desktop como la versión Server, estará disponible a partir del día 29 de abril de 2010.
Para distribuciones LTS (Long term support) de Ubuntu, Canonical provee ayuda técnica y actualizaciones durante 3 años para la versión escritorio y 5 años para la versión servidor, a partir de la fecha del lanzamiento.

Ubuntu Server es la distribución Linux utilizada en todos los servidores del departamento, es esta la razón y no otra por la que se ha elegido esta distribución, el trabajar con una versión ya conocida supone cierta comodidad y amigabilidad para los administradores de SIETCG. Además, solo se instalan distribuciones LTS por su amplio soporte, ya que gracias a este soporte no es necesario estar haciendo actualizaciones del servidor cada pocos meses.

Para instalar Openfire se requiere instalar el conjunto de software del paquete LAMP Server ya que son necesarios los paquetes, Linux, Apache y MySQL.
De cara a facilitar el acceso remoto al servidor es necesario instalar el paquete OpenSSH server. Con este paquete podremos acceder vía Secure Shell desde desde cualquier estación de administración.

En el documento técnico de instalación “DT.I.01 Instalación Ubuntu server 8.04 LTS”, se describen los pasos para la instalación de un servidor Linux con Ubuntu y adecuado para una instalación de Openfire.

Además, en este servidor se han instalado los paquetes build-essentials y Linux headers, el paquete build-essentials contiene las herramientas necesarias para compilar programas, requerido por ejemplo para instalar las VMware tools que también hemos instalado, además, para compilar VMware tools, se requiere del código fuente del kernel (Linux-headers).

Las VMWare tools son un paquete de drivers que hacen que la máquina virtual se comunique con su host (el servidor ESXi), y que proporcionan drivers para los distintos dispositivos hardrware virtuales (VGA Virtual, USB virtual, controlador de red y de disco virtual, etc).
Finalmente también se han instalado los componentes cliente de Nagios para monitorizar la máquina. Los componentes instalados son nagios-nrpe y nagios-plugins. La idea es dejar preparada la instalación para poner en marcha la monitorización de la misma, cuando entre en explotación. Se monitorizarán los discos, carga de trabajo y si hay o no actualizaciones disponibles.

5.4.3 Instalación de Openfire:

A fecha de octubre de 2009, la última versión disponible de Openfire que encontramos es la 3.6.4 y está disponible desde el 1 de mayo de 2009. Por el momento en el roadmap de Igniterealtime no se describen planes futuros para la disposición de nuevas versiones. Por lo tanto, la versión de Openfire a instalada en ETCG es la 3.6.4.

El proceso de instalación de Openfire está dividido en las siguientes fases:

- Instalación de la máquina virtual Java.

- Instalación de Openfire.
- Creación y configuración base de datos MySQL.
- Configuración o Setup de Openfire.

Es requisito explicito de Openfire, la instalación de la máquina virtual Java para el funcionamiento de Openfire. En este caso, también se ha instalado la última versión disponible que es la versión 6, update 17.
Openfire necesita una base de datos en la que almacenará toda la información, tanto datos de configuración como datos de estado y registro de usuarios. Esta base de datos puede ser interna o externa.
La base de datos interna de Openfire esta implementada con HSQLDB, se trata de una base de datos construida en Java. El uso de esta base de datos no está recomendado para entornos de producción, por lo tanto, igual que se ha hecho en el entorno del laboratorio virtual, se ha decidido usar una base de datos externa, concretamente una base de datos implementada con MySQL.

En el Setup de Openfire se ha tenido en cuenta la integración de Openfire con el directorio activo del departamento.
Para esta integración desde SIETCG se facilitó la siguiente información:

Host:	nala.upc.es
Base DN:	OU=Usuarios Normales, OU=etcg, DC=upc, DC=es
Administrator DN:	etcg\admin_im
Password:	*********

Esta integración permite que los usuarios de ETCG utilicen el mismo usuario y contraseña que utilizan para validarse en el dominio y por otro lado, reduce el trabajo del personal de SIETCG, ya que ni es necesario el alta de usuarios ni el mantenimiento de los mismos desde Openfire, este trabajo queda centralizado en el directorio activo del departamento.

La instalación de Openfire se ha hecho según lo descrito en el documento técnico de instalación "DT.I.02 Instalación Openfire 3.6.4".

5.4.4 Instalación servicios valor añadido:

Una vez instalado y probado Openfire, se instalaron los servicios de valor añadido.
Entre los presentados ante SIETCG, se seleccionaron los siguientes:

- Implemente videollamada.
- Acceso a la plataforma de mensajería a través de un cliente web.
- Acceso a plataformas IM externas.

Para implementar videollamada se ha recurrido al Plugin Red5 de Openfire. Las características de este plugin están recogidas en el apartado 4.5 de este documento y su instalación queda descrita en el documento técnico de instalación, "DT.I.11. Instalación Plugin Red5".

Para implementar el acceso a la mensajería instantánea a través de un cliente web, se ha recurrido al cliente Sparkweb. Para la instalación de este software aprovecharemos el propio Apache que tenemos instalado en el servidor de Openfire y no necesitaremos ningún servidor web externo.

Al cliente web se accede a través de cualquiera de estos enlaces:

http://imetcg.upc.es/sparkweb/SparkWeb.html o directamente a través de http://imetcg.upc.es

La instalación de este software queda descrita en el documento técnico de instalación, “DT.I.03 Instalación cliente SparkWeb”.

Para el acceso remoto, se ha requerido a SIETCG acceso a los siguientes puertos:

Puerto:	Descripción:
80	Acceso web.
5222	Cliente Jabber.
9090	Administración Openfire.
843 y 5229	Acceso cliente web.
59590, 59591 y 59592	Videollamada.

También se ha requerido acceso a través del alias http://imetcg.upc.es

Para el acceso a plataformas de mensajería instantánea externa, tales como MSN o Google Talk, se ha recurrido a la instalación del plugin Kraken.

5.4 Monitorización de Openfire:

La monitorización de la plataforma de mensajería instantánea se hace mediante la plataforma Nagios.
Nagios es un sistema open source de monitorización de redes ampliamente utilizado, que monitoriza a los equipos (hardware) y servicios (software) especificados, alertando cuando el comportamiento de los mismos no sea el deseado. Entre sus características principales figuran la monitorización de servicios de red (SMTP, POP3, HTTP, SNMP,etc), la monitorización de los recursos de sistemas hardware (carga del procesador, uso de los discos, memoria, estado de los puertos, etc), e independencia de sistemas operativos.

Se trata de un software que proporciona una gran versatilidad para consultar prácticamente cualquier parámetro de interés de un sistema, y genera alertas, que pueden ser recibidas por los responsables correspondientes mediante (entre otros medios) correo electrónico y mensajes SMS.

De la puesta en marcha de la monitorización, por políticas de seguridad, se han encargado los técnicos de SIETCG. Para esta monitorización, en el servidor de Openfire se ha instalado el nagios-nrpe-server, nagios-nrpe-plugins y nagios-plugins. Se ha modificado la configuración en /etc/nagios para dar acceso de consulta al servidor Nagios, y se ha creado una instancia en el server nagios.

Ahora, desde la plataforma de monitorización Nagios, junto con el resto de servidores del departamento, se esta monitorizando Openfire:

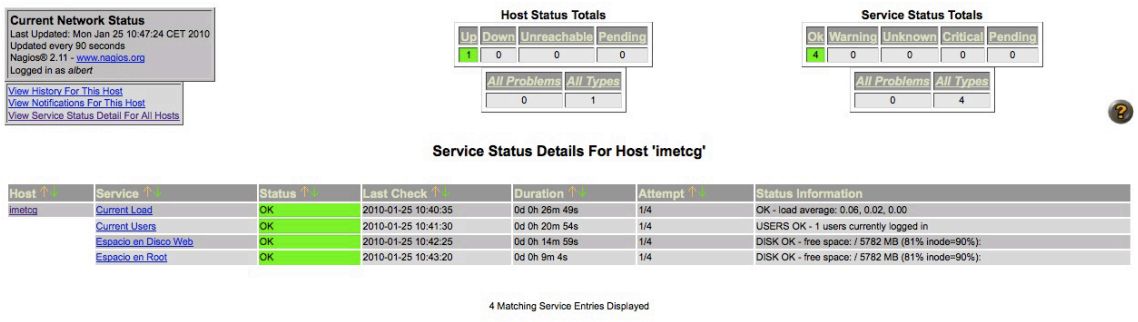


Figura 68. Monitorización Nagios.

5.5 Personalización cliente Spark para ETCG:

La personalización del cliente de mensajería instantánea no era un requerimiento inicial, sino que se consideró importante la expansión de la imagen corporativa del departamento en el propio software corporativo.

La personalización de este software ha sido posible gracias a los ventajas de trabajar con software Open Source. El código de todas las versiones del cliente Spark está disponible en este servidor de Subversión: http://svn.igniterealtime.org/svn/repos/spark/tags

Puesto que se supone que la última versión del cliente Spark, es la que corrige más bugs e implementa más funcionalidades, ha sido esta la versión con la que se ha trabajado, la versión 2.5.8.

El cliente de mensajería instantánea se ha personalizado con el logotipo del departamento.
En la siguiente imagen se muestra la personalización hecha en el cliente Spark:

Figura 69. Personalización cliente Spark.

Para la descarga del código fuente de Spark ha sido necesaria la instalación de un software de gestión de subversiones, el software seleccionado ha sido "tortoiseSVN", se trata de un software gratuito y bajo licencia GNU GPL.
Este software se ha descargado desde la propia página web del fabricante,
http://tortoisesvn.net/downloads

Una vez editado el código fuente para incluir el nuevo logotipo, ha sido necesario compilar este código. El código fuente de Spark está escrito Java, para su compilación se ha usado el software: NetBeans IDE 6.8. Esta aplicación requiere del "Java Developer Kit", ambos software se pueden instalar juntos a través del paquete "jdk-6u17-nb-6_8-windows-ml.exe" y se puede descargar directamente de la página web de sun microsystems,
http://java.sun.com/javase/downloads/widget/jdk_netbeans.jsp

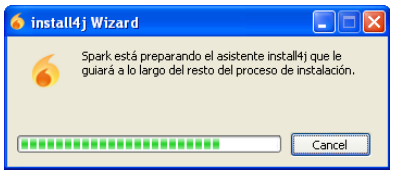
Para tener un instalable con el código modificado es necesario tener un software capaz de generar este instalable. El software seleccionado ha sido "install4j" de EJ-Technologies, se trata de un wizard que partiendo del código fuente compilado y personalizando una serie de opciones acaba generando un fichero ejecutable.
A diferencia del software anterior, este último no es gratuito, pero desde la página web del fabricante se ha podido descargar una versión de evaluación de 90 días. Esta versión de

evaluación se ha descargado de este link:
http://www.ej-technologies.com/download/install4j/files.html

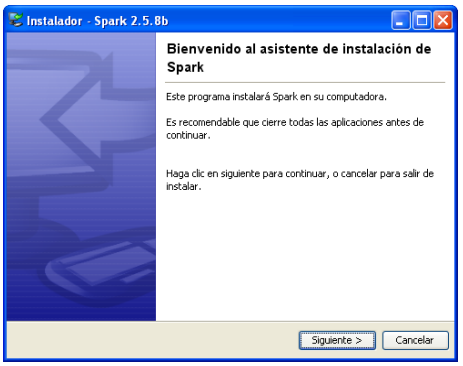
El instalable generado mantiene el logotipo original: 

El wizard de instalación del paquete original era en Ingles, en el paquete personalizado se ha cambiado y se ha creado en castellano.
Al arrancar el wizard de instalación aparecen las siguientes ventanas:

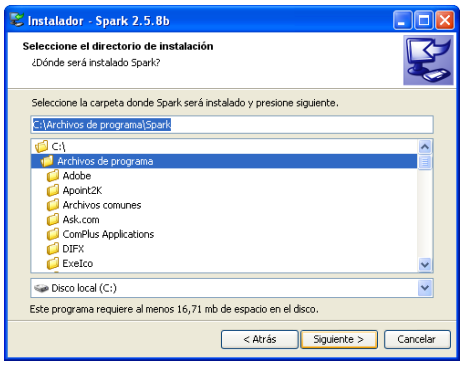
1.



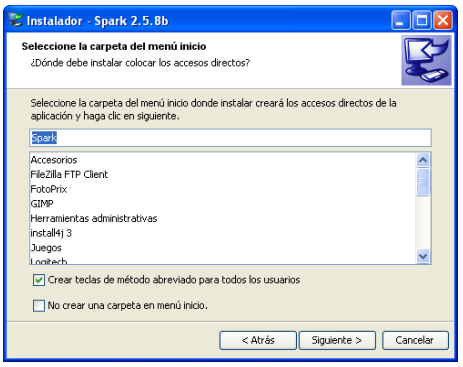
2.



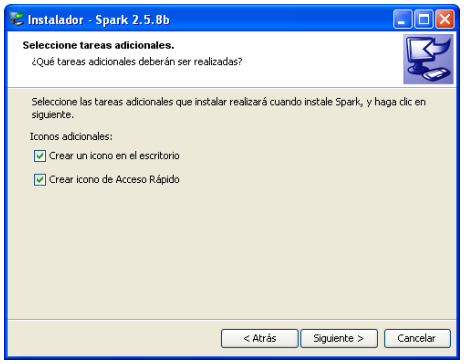
3.



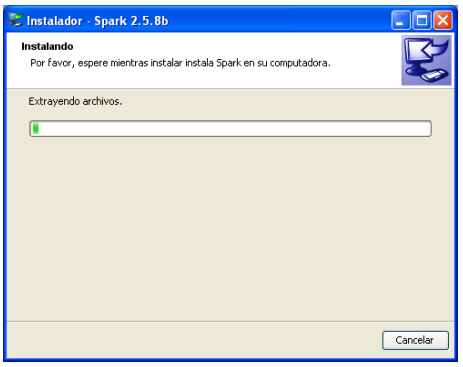
4.



5.



6.



7.

Figura 70. Instalación Spark en ETCG.

Salvo lo comentado sobre el idíoma del wizard de instalación, el resto de la instalación es idéntica a la del paquete Spark original.

Tras la instalación se han crado los siguientes icónos,

Sobre la barra de tareas:

Sobre el escritorio:

Figura 71. Iconos instalación Spark en ETCG.

Tras la exitosa instalación, también se ha validado que el funcionamiento sea el mismo que el del cliente Spark original.

En este caso solo se ha personalizado el logotipo de bienvenida, pero editando el código fuente se podría haber modificado completamente el aspecto visual del cliente e incluso sus funcionalidades.

Con esta personalización se abre un nuevo punto de vista para la personalización del cliente de mensajería instantánea, cuando se distribuya e instale esta aplicación, de forma corporativa en toda la UPC.

En el documento técnico de instalación, “DT.I.15 compilación y generación instalable del cliente Spark”, quedan descritos los pasos seguidos para la compilación y generación del nuevo instalable.

5.6 Distribución Software y manuales usuarios:

Desde SIETCG se encargarán de hacer la distribución del software y de los manuales de usuario. Esta distribución se hará a través de la unidad de software del departamento y a través de la Intranet.

La instalación del cliente Spark correrá a cargo de los propios usuarios.

5.7 Acceso externo:

Desde SIETCG se nos ha facilitado acceso a través de la dirección IP pública 147.83.51.222 a los puertos solicitados.

SIETCG se puso en contacto con ATIC[21] para solicitar una entrada de host en el DNS de la UPC, de tal forma que el alias imetcg.upc.es resolviera con la IP pública que nos habían facilitado.

Siendo así, esta plataforma se podrá administrar externamente a través del enlace, http://imetcg.upc.es:9090 o acceder a su cliente web directamente a través del enlace, http://imetcg.upc.es.

[21] ATIC, Centro de **A**tención **TIC**, atención a los usuarios de servicios de UPCnet.

6. Planificación del proyecto

En las fases iniciales de este proyecto se hizo una planificación temporal y económica.
La planificación de un proyecto es una herramienta muy útil que permite estimar plazos y realizar un seguimiento del desarrollo del proyecto, por lo que nos permitirá detectar desviaciones y poner de manifiesto las carencias no contempladas en las fases iniciales y por lo tanto, actuar en consecuencia para corregir estas desviaciones.

6.1 Planificación temporal:

La planificación inicial de este proyecto ha sufrido ciertas desviaciones respecto a lo previsto inicialmente. Estas desviaciones han sido debidas básicamente a que han variado los objetivos de este PFC. A la hora de ir conociendo Openfire y descubriendo sus capacidades, se ha decidido ampliar el estudio del abanico de posibilidades que nos ofrece. Por lo tanto, estas variaciones han supuesto modificaciones temporales que han retrasado unas semanas la finalización de este proyecto.

Además de la variación temporal debido a la ampliación del estudio y ensayo de las diferentes posibilidades que ofrece Openfire, en alguno de los puntos de estudio, debido a que ha supuesto mayor dificultad de lo esperado, también a supuesto el consecuente incremento de tiempo hasta la resolución de dichas dificultades.

La planificación de este proyecto se ha dividido 5 fases:

1. Propuesta PFC y definición de objetivos, al tratarse de una colaboración con UPCnet, se mantuvieron varias reuniones iniciales en las que se hizo una propuesta de colaboración y en base a las necesidades expuestas se marcaron los objetivos de este proyecto.

2. Análisis, se realiza un estudio de viabilidad revisando las posibles soluciones técnicas que se pueden ofrecer. Una vez revisada la viabilidad se dedica cierto tiempo al estudio de protocolos que pueden ser necesarios para una implementación técnica y se decide que plataforma será la que finalmente se tome como solución técnica a estudio e implementación. Para el estudio de esta solución técnica, se monta un entorno de laboratorio virtual de pruebas.

3. Diseño, se estudian y testean todas aquellas posibilidades que ofrece Openfire y que en la implementación de una solución global de comunicaciones unificadas pueden resultar interesantes.

4. Implementación, tras varias reuniones entre UPCnet y Departamento de Ingeniería del Terreno, Cartográfica y Geofísica de la UPC, se decide hacer una implementación piloto en este Departamento. La implementación piloto esta adecuada a los requerimientos propuestos por ETCG.

5. Documentación del proyecto, se documenta tanto el estudio realizado como los procedimientos de instalación y manuales de usuario y administración.

La dedicación a este proyecto ha sido de 3 horas diarias de lunes a viernes (de 18h a 21h) y 3 horas los sábados (de 10h a 13h). Excepcionalmente se ha dedicado tiempo fuera de estas horas para reuniones y para la instalación y puesta en marcha de la plataforma piloto de pruebas.

La asignación de recursos humanos en este proyecto está dedicada prácticamente en exclusiva al técnico encargado del desarrollo del mismo, la única excepción ha sido la dedicación de uno de los técnicos de SIETCG para la puesta en marcha de la máquina virtual, la instalación del SO donde se instaló Openfire, la puesta en marcha de la monitorización con Nagios y la gestión del acceso externo a la plataforma.
En este proyecto, dado que la mayoría de las tareas han sido ejecutadas por el mismo técnico, no se han podido solapar tareas.

En la siguientes dos páginas se anexa el diagrama de Gantt donde se ha plasmado toda la planificación temporal del proyecto. Este diagrama se ha fraccionado en dos páginas para una correcta visualización.

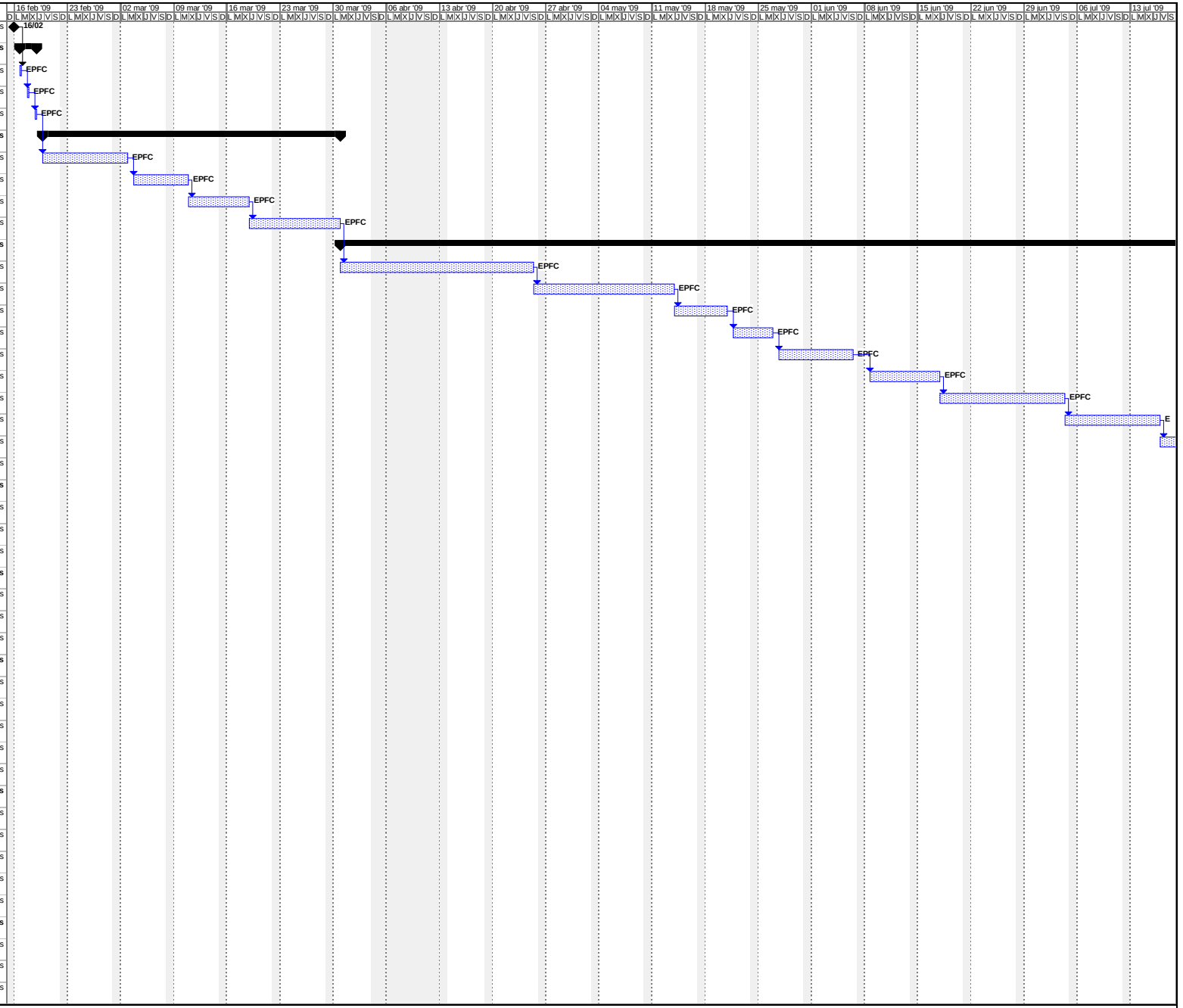
En el diagrama de Gantt se representan los siguientes recursos:

- EPFC: Estudiante Proyecto Final de Carrera.
- SIETCG: Sistemas Informáticos ETCG.

En la planificación de este proyecto se han establecido los siguientes periodos no laborables:

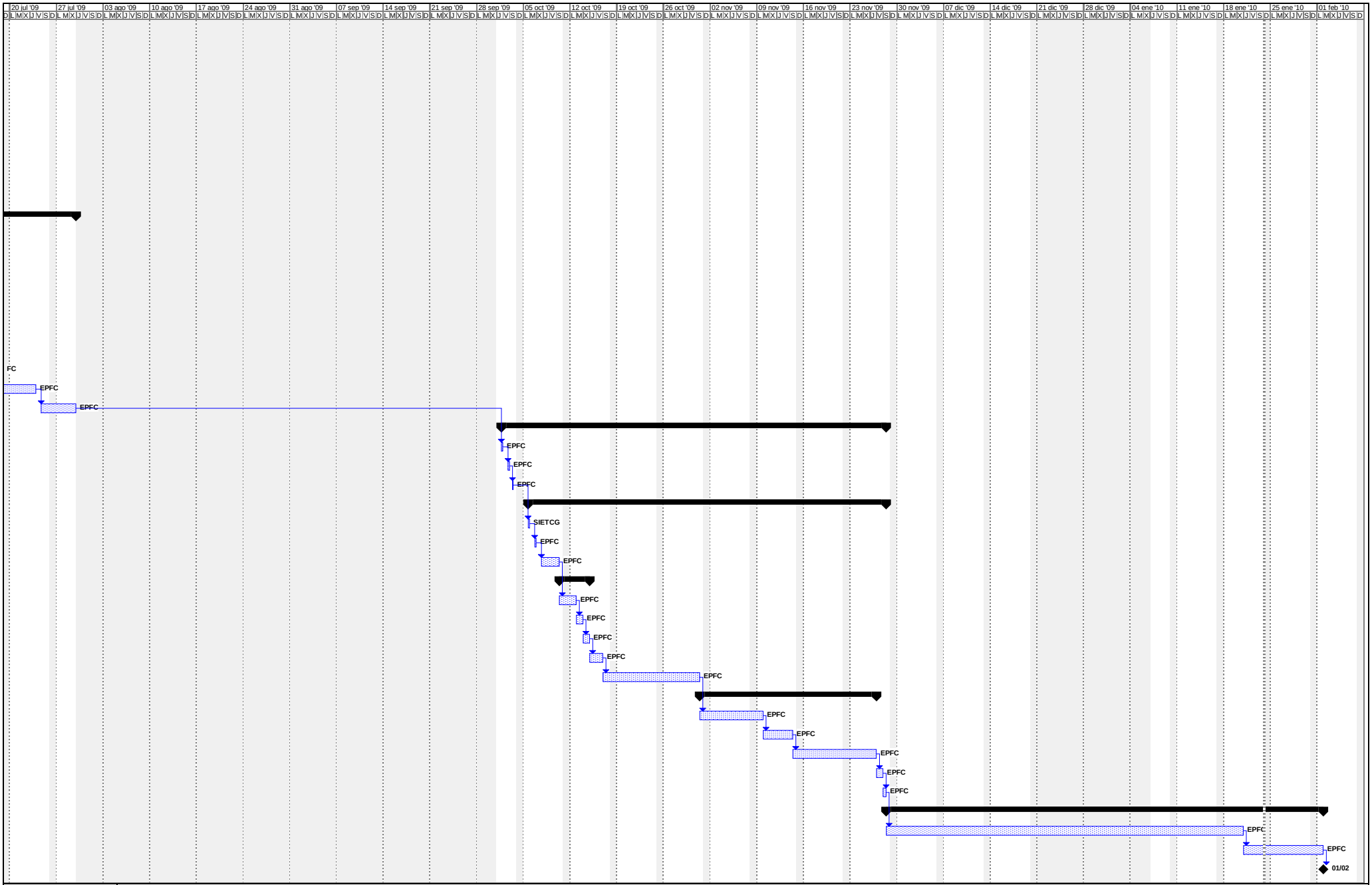
- Semana Santa 2009: del 4 al 13 de abril.
- Vacaciones estivales: del 30 al 31 de agosto.
- Baja por paternidad: del 1 al 30 de septiembre.
- Vacaciones Navidad: del 21 de diciembre al 6 de enero.

Id	Nombre de tarea	Duración
1	Inicio del Proyecto.	0 horas
2	**Propuesta PFC y definición objetivos:**	**3 días**
3	Reunión UPCnet, propuesta Proyecto.	3 horas
4	Reunión UPCnet, definición de objetivos.	3 horas
5	Reunión con el ponente del PFC.	3 horas
6	**Análisis de la situación a resolver:**	**33,33 días**
7	Revisión soluciones.	30 horas
8	Estudio conceptos previos, protocolo SIP.	20 horas
9	Estudio conceptos previos, protocolo XMPP.	20 horas
10	Implementación Laboratorio de pruebas.	30 horas
11	**Diseño plataforma:**	**96,67 días**
12	Pasarela XMPP entre las plataformas de VoIP e IM.	45 horas
13	Traspaso de presencia entre las plataformas de VoIP e IM.	45 horas
14	Gestión de usuarios a través de OpenLDAP.	20 horas
15	Integración usuarios SIP en clientes Jabber.	15 horas
16	Implementación video llamada desde Openfire.	30 horas
17	Implementación cliente Sparkweb.	25 horas
18	Alta disponibilidad y alto rendimiento.	45 horas
19	Servicios de web collaboration para centro de contactos.	30 horas
20	Escalabilidad de Openfire.	20 horas
21	Pasarela con otras plataformas de mensajería.	15 horas
22	**Implementación plataforma piloto:**	**50,33 días**
23	Reunión UPCnet, revisión estudio y propuesta piloto en ETCG.	3 horas
24	Reunión SIETCG, presentación funcionalidades Openfire.	3 horas
25	Reunión SIETCG, toma de requerimientos y presentación red ETCG.	3 horas
26	**Instalación Openfire en ETCG:**	**47,33 días**
27	Instalación máquina virtual en ETCG.	3 horas
28	Instalación SO.	3 horas
29	Instalación Openfire.	10 horas
30	**Instalación servicios de valor añadido:**	**3 días**
31	Video llamada.	3 horas
32	Acceso a cliente web.	3 horas
33	Pasarela con otras plataformas de IM.	3 horas
34	Monitorización Openfire.	6 horas
35	Personalización cliente Spark para ETCG.	40 horas
36	**Documentación:**	**21,67 días**
37	Manual usuario cliente Spark.	20 horas
38	Manual usuario cliente Sparkweb.	15 horas
39	Manual administración Openfire.	30 horas
40	Reunión SIETCG, entrega documentación.	3 horas
41	Reunión SIETCG, formación administradores.	3 horas
42	**Documentación del Proyecto:**	**40 días**
43	Elaboración de la documentación del PFC.	90 horas
44	Elaboración presentación PFC.	30 horas
45	Fin del Proyecto.	0 horas



Proyecto: PFC Jose Luis Villacam
Fecha: dom 24/01/10

Tareas críticas
División crítica
Progreso de tarea crítica
Tarea
División
Progreso de tarea
Línea de base
División de la línea de base
Hito de línea de base
Hito
Progreso del resumen
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Fecha límite


20 jul '09
27 jul '09
03 ago '09
10 ago '09
17 ago '09
24 ago '09
31 ago '09
07 sep '09
14 sep '09
21 sep '09
28 sep '09
05 oct '09
12 oct '09
19 oct '09
26 oct '09
02 nov '09
09 nov '09
16 nov '09
23 nov '09
30 nov '09
07 dic '09
14 dic '09
21 dic '09
28 dic '09
04 ene '10
11 ene '10
18 ene '10
25 ene '10
01 feb '10
FC
EPFC
SIETCG
01/02
Proyecto: PFC Jose Luis Villacam
Fecha: dom 24/01/10
Tareas críticas
División crítica
Progreso de tarea crítica
Tarea
División
Progreso de tarea
Línea de base
División de la línea de base
Hito de línea de base
Hito
Progreso del resumen
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Fecha límite

6.2 Planificación económica:

Se ha hecho una planificación económica basándose en las tareas de implementación de Openfire, no se ha cuantificado la parte de investigación y documentación de este proyecto.
Para la planificación económica de este proyecto se han cuantificado los costes humanos y costes hardware.

En cuanto a los costes software en licencias son de cero euros. El desarrollo de este proyecto se ha basado en software libre, siguiendo la política de uso de software libre promovida tanto por UPCnet como por la propia UPC.

Respecto al dimensionamiento y coste de la máquina donde se aloja Openfire, tuvimos mucha suerte porque a la hora de implementar Openfire, coincidió con la puesta en marcha de un servidor donde se alojan otros 9 servicios más y cuyo coste ya estaba planificado para este otro proyecto, de todas formas a la hora de cuantificar costes y que además sirva de referencia para UPCnet y para el propio ETCG a la hora de una futura renovación del equipo, se ha cuantificado como si también se compartieran gastos.
La máquina se ha comprado con una garantía de 3 años con el fabricante y una extensión de 5 años más a través un convenio que tiene la UPC con Bull. De todas formas, está previsto un tiempo de vida tecnológico para este hardware de 5 años. Por lo tanto, el plazo de amortización para este hardware es de 5 años.
El coste total de este equipo ascendió a 6.972,45€.
Si además tenemos en cuenta que el coste de este servidor se ha de repartir entre los siguientes servicios:

Servicio:	Descripción:
Simba	Controlador de dominio SIETCG.
VPNEtcg	Servidor de acceso remoto al departamento.
WebMIVES	Servidor web de www.mives.upc.es
VMWare Admin	Estación de trabajo de acceso remoto para administrar el VMWare.
WebLLISCAT	Servidor web de www.lliscat.upc.es
WebPRH2O	Servidor web de www.proyectosh2o.upc.es
Web2etcg	Servidor web de www2.etcg.upc.es
Webhidrologia	Servidor web de www.h2ogeo.upc.es
SubVGHS	Servidor de Subversión para el grupo de hidrología subterránea.
IMETCG	Servidor mensajería instantánea.

El coste resultante para el servidor de mensajería instantánea es de 697,245€

Para el cálculo de costes humanos, se han tenido en cuenta las horas de ingeniería invertidas en la puesta en marcha de la plataforma en ETCG, incluidas las horas de invertidas en la documentación de manuales técnicos, reuniones y personalización del cliente Spark, esta personalización solo incluye la introducción del logo del Departamento, no el desarrollo de nuevas funcionalidades.
En total se han contabilizado 151 horas. El coste de las horas de ingeniería asciende a 39,12 €/hora. Esta tarifa esta extraída del convenio colectivo del gremio de las consultoras y corresponde a las tarifas de ingeniería de un técnico Titulado Superior.
El total de las horas de ingeniería asciende a: 5907,12€.

Se considera que esta plataforma requiere un total de 20 horas anuales de mantenimiento dedicadas a la actualización software de la plataforma, administración y mantenimiento hardware de la misma.

El coste de este mantenimiento a lo largo de los cinco años de vida de la plataforma, asciende a: 3.912€.
El resumen de costes de implantación y mantenimiento de la plataforma extendido sobre los cinco años de vida de la misma, es el siguiente:

COSTES HUMANOS:		
Concepto:	**Detalle:**	**Coste:**
Horas Ingeniería.	151 h * 39,12 €/h	**5.907,12 €**
Mantenimiento (5 años).	20 h/año * 5 años * 39,12 €/h	**3.912 €**
COSTES MATERIALES:		
Concepto:	**Detalle:**	**Coste:**
Hardware.	Servidor Dell PowerEdge	**697,245**
	TOTAL:	**10.516,365 €**

Considerando que el tiempo de vida de estimado para este servidor es de 5 años y que este servidor proveerá de servicio de mensajería instantánea a 250 usuarios, se puede calcular un coste por usuario y por mes:

$$\frac{\frac{10.516{,}365\ €}{250\ \text{Usuarios}}}{5\ \text{años}*\frac{12\ \text{meses}}{1\ \text{año}}} = 0{,}7011€(usuario/mes)$$

Los costes de este servidor son muy bajos para todos beneficios de los servicios de valor añadido que suponen.

7. Conclusiones y líneas de futuro

7.1 Conclusiones:

La investigación realizada ha permitido por un lado, conocer con detalle una plataforma de mensajería instantánea, revisar algunas de las funcionalidades de valor añadido que presenta y pueden ser interesantes para la implantación en el sistema global de comunicaciones unificadas de la UPC y recopilar la documentación técnica de instalación y manuales técnicos de usuario que pueden ser de gran ayuda para UPCnet en la puesta en marcha e implantación de esta plataforma.

Por otro lado, se ha hecho una implementación piloto en el Departamento de Ingeniería del Terreno, Cartográfica y Geofísica de la UPC. A esta plataforma se tiene acceso externo directamente desde Internet, sin establecer ninguna conexión VPN previa, esta sería la foto final de una plataforma global de comunicaciones unificadas, en las que todo el mundo pudiera acceder desde cualquier parte y desde cualquier terminal con conexión a Internet.
Actualmente en la fase de escritura de este trabajo, los técnicos de SIETCG están testeando la nueva aplicación en un grupo reducido de usuarios, pero tras las validaciones previas esta previsto distribuir el software a todos los miembros del Departamento y reforzar la seguridad de la red con el exterior, desde los PCs de los usuarios quedará bloqueado el acceso a las plataformas de mensajería instantánea públicas, solo podrán acceder a estas a través de Openfire y se dará acceso a aquellos usuarios que lo soliciten y estén dentro de las políticas de accesibilidad del Departamento.

Esta implementación piloto, permitirá a UPCnet conocer la experiencia de los usuarios, con sus puntos fuertes y sus carencias para intentar resolverlas si es que se presentaran.

En las fases de estudio e implementación en el entorno de laboratorio, las fases que mayor dificultad han supuesto, han sido las fases de integración con la plataforma de VoIP Openser, concretamente conseguir el traspaso de presencia y de mensajería instantánea entre ambas plataformas ha sido quizás lo más complicado ya que ha supuesto conocer el funcionamiento de los protocolos SIP y XMPP y el detalle de los módulos de integrabilidad de Openser y Openfire. Esta inversión de tiempo ha valido la pena ya que los resultados han sido fructíferos y ha permitido cumplir con uno de los objetivos principales y de los que mayor interés suponía para UPCnet.

Personalmente el colaborar con una organización como UPCnet en la elaboración de este estudio y en la implantación del piloto de pruebas, me ha permitido conocer la organización, la metodología de trabajo y las líneas transversales de colaboración entre UPCnet y los Departamentos de la UPC, en este caso con ETCG.
Este trabajo, me ha permitido en cierta forma gestionar los recursos humanos de UPCnet y ETCG cuando he requerido su colaboración, en momentos puntuales del desarrollo del mismo y cuando solo era posible la continuidad del trabajo canalizando alguna tarea bien a través de UPCnet, sobre todo en la fase de estudio, o bien a través de ETCG, en este caso sobre la fase de implementación.
El desarrollo de este trabajo también me a aportado el conocimiento de protocolos SIP y XMPP y además me ha permitido conocer una plataforma de mensajería Open Source sobre un SO Linux, con lo que también he podido familiarizarme con este tipo de sistemas operativos que eran muy poco conocidos para mí y que actualmente la industria esta haciendo proliferar mediante plataformas basadas en RTOS Linux, y por lo tanto son de valorar los conocimientos adquiridos.

Los objetivos marcados al inicio de este proyecto se han cumplido exhaustivamente, UPCnet podrá disponer de la documentación necesaria que les permitirá implantar esta plataforma de forma guiada o bien en el caso que finalmente la nueva plataforma se implante por un "Third Party Installer", permitirá hacer un seguimiento por parte de UPCnet en las fases de instalación y posterior mantenimiento.

Durante este trabajo ha ido variando la planificación, debido a la dificultad en el estudio y resolución de alguno de los puntos y dado que finalmente se han estudiado más funcionalidades de valor añadido de las que inicialmente se había pensado. Estas modificaciones no han variado los objetivos principales, pero si que han ampliado el abanico de funcionalidades estudiadas ya que resultan de gran interés para UPCnet.

7.2 Líneas de futuro:

La aparición constante de nuevos plugins desarrollados por la comunidad de Openfire hace que vayan apareciendo nuevas funcionalidades que pueden ser útiles y de gran interés para cualquier plataforma de comunicaciones unificadas. Pero al margen de estos plugins que pueden ir apareciendo y de cara a la implantación final en la UPC, hay varios puntos que pueden ser de gran interés, estos son los siguientes: Resolución de la redundancia con plataformas de virtualización VMware, integración de la plataforma de gestión de llamadas del Call Center con Openfire y paquete Fastpaht y también puede ser interesante para por ejemplo un entorno como ETCG que tiene a Nagios como herramienta de monitorización, la recepción de alertas vía mensajería instantánea.

Actualmente esta proliferando la virtualización de sistemas mediante plataformas VMware, concretamente la plataforma VMware ESX permite la creación de clusters de servidores de tal forma que en caso de caída o pérdida de visibilidad con uno de los nodos, otro de ellos asume la carga de trabajo de forma transparente por los usuarios. Esto vendría a ser lo mismo que se puede conseguir con el plugin Clustering de Openfire, pero con esta metodología de clustering se libera el consumo de recursos de Openfire, ya que una configuración en clustering supone una carga adicional, y queda bajo el control de VMware. Como contrapartida tenemos que la versión VMware ESX no es gratuita, pero si bien es cierto, esta es la distribución más usada por los departamentos de IT de la escuela.

Desde el Call Center de ATIC se atienden contactos de llamadas entrantes. Con una pequeña integración entre Openfire y la plataforma de gestión de llamadas, se podría ampliar el abanico del centro de atención para también poder atender consultas Chat. Esto sería muy útil sobre todo para aquellos alumnos que en estancias de Erasmus necesitaran contactar desde el extranjero con ATIC, ahorrarían costes en llamadas internacionales.
La idea de esta integración es la siguiente, una vez habilitado el paquete Fastpaht y establecidas las reglas de enrutamiento de contactos chat, debería desarrollarse una integración que permitiera al software de gestión de llamadas conocer el estado de los agentes del call center, es decir, que se alimentara de las tablas de Presencia para conocer si el agente esta libre u ocupado atendiendo algún contacto de IM o chat y a su vez, cada vez que se entregara una llamada a un agente, esta integración actualizará el estado de presencia para evitar que cuando se este atendiendo una llamada entrará de forma simultanea un contacto de Chat.

A continuación podemos ver un esquema del funcionamiento de la integración entre Openfire y el software de gestión telefónica que permitirá gestionar contactos de Voz y contactos Chat de forma paralela y ordenada, de tal forma que un agente no recibirá de forma simultánea ambos tipos de contacto.

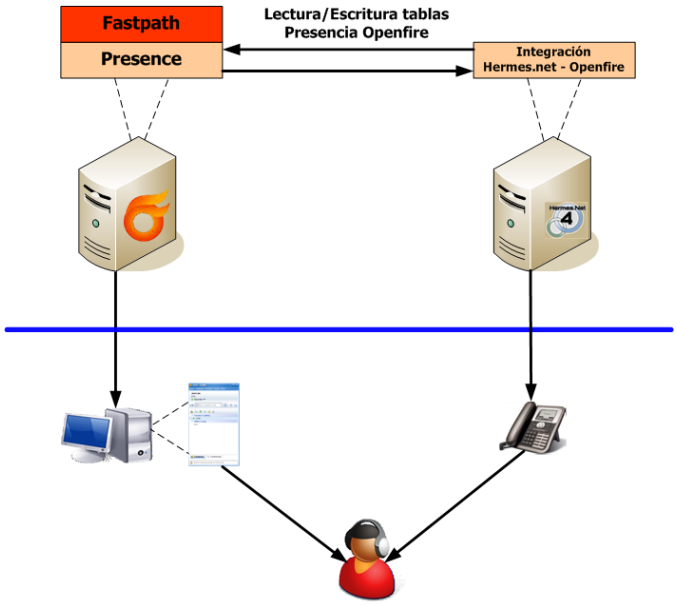


Figura 72. Integración Contact Center con Openfire.

Cuando en una organización se monitorizan servidores u otros equipos, una rápida respuesta muchas veces evita la pérdida de algún servicio y el consecuente impacto sobre los usuarios. Para tener una monitorización eficiente, es imprescindible que las notificaciones sean en tiempo real. Muchas veces en el día a día de muchas organizaciones no se puede disponer de un equipo de personas encargadas de estar pendientes de las consolas de monitorización y activar los procedimientos de recuperación de forma instantánea, de tal forma que a veces el departamento de IT, recibe la incidencia por parte de algún usuario antes de percatarse de la alarma en la consola de monitorización.
Por ejemplo, para una organización como el Departamento de ETCG que dispone de monitorización a través de Nagios y que ya dispone de una plataforma de mensajería instantánea con Openfire, es posible enviar las notificaciones a los técnicos de SIETCG a través de Openfire vía mensaje instantáneo.

En la siguiente imagen se ve un ejemplo de notificación vía mensajería instantánea con Openfire y su cliente Spark:

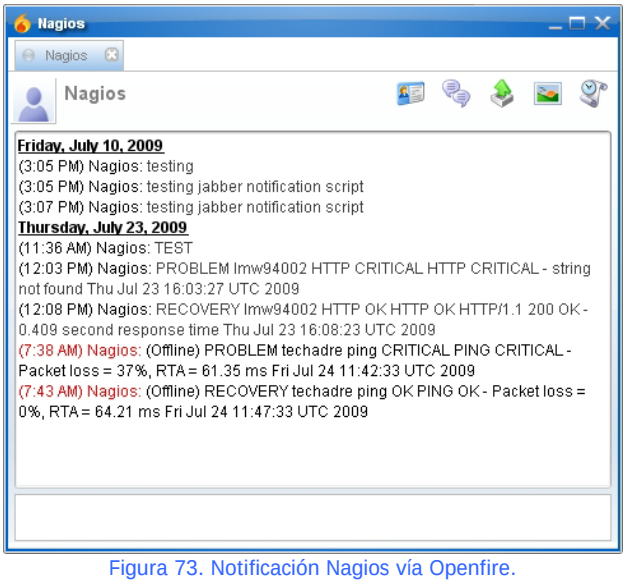


Figura 73. Notificación Nagios vía Openfire.

Anexo técnico

Este anexo técnico esta dividido en dos partes, en una primera parte se recoge a modo de procedimientos los documentos técnicos que podrán guiarnos para la instalación de Openfire y de cualquiera de sus servicios de valor añadido. Estos documentos están descritos con la nomenclatura DT.I.xx.

En una segunda parte se recogen los manuales de usuario de Openfire y sus guías rápidas. Estos documentos están descritos como DT.U.xx.

Documentos técnicos de instalación:

DT.I.01. Instalación Ubuntu Server 8.04.10 LTS:

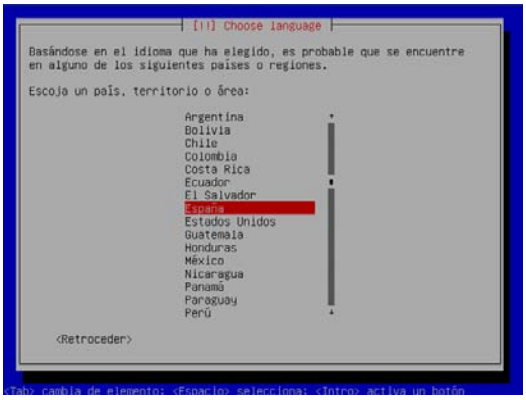
Desde el portal web de Ubuntu nos podemos descargar la imagen con la distribución de Ubuntu server que necesitamos.

Una vez descargada la grabamos en un CD en blanco y seguimos con los siguientes pasos:

1. Insertamos el CD en el lector y arrancamos el Servidor o la máquina virtual donde se vaya a instalar:

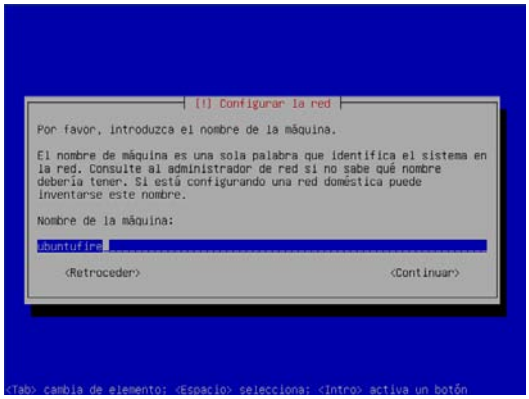
DTI. Fig 1 Pantallas instalación Ubuntu 8.04.

2. Seleccionamos el Lenguaje y el país donde nos ubicamos:



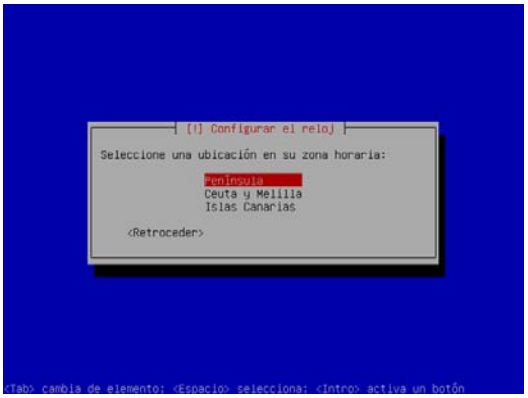
DTI. Fig 3.

3. Configuración del Hostname del servidor:



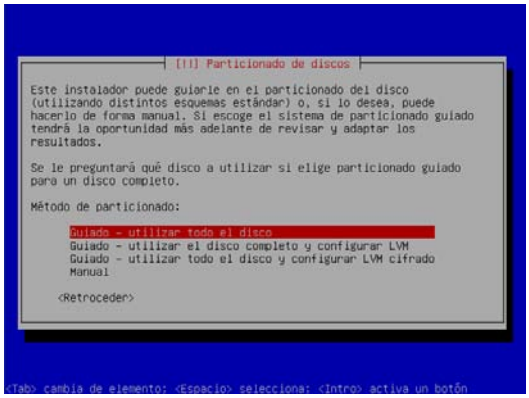
DTI. Fig 2.

4. Configuración de la Zona horaria:



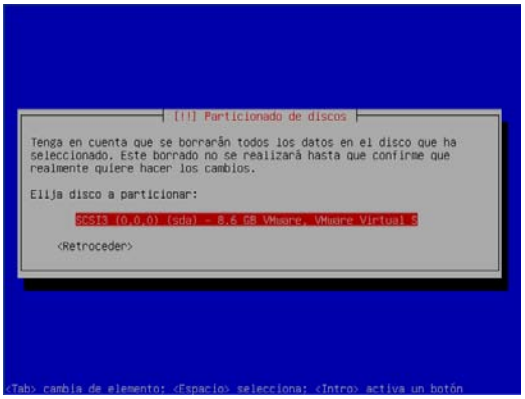
DTI. Fig 4.

5. Inicio particionado del disco:



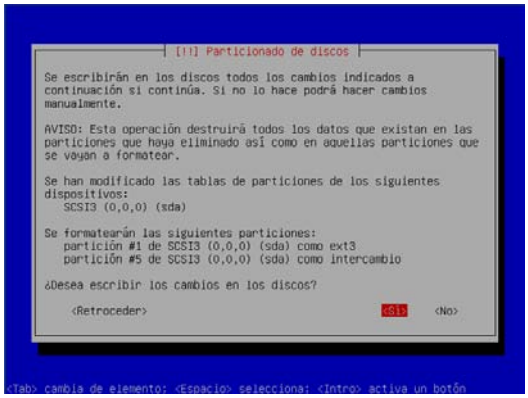
DTI. Fig 5.

6. Selección disco a particionar:



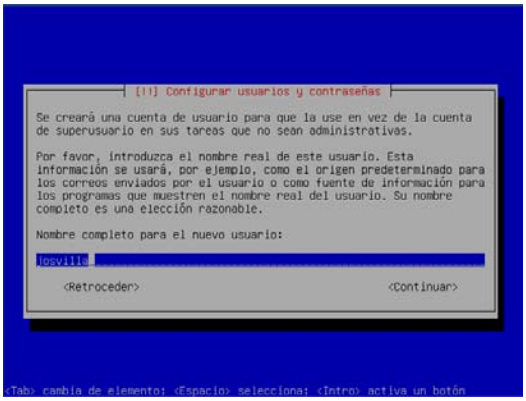
DTI. Fig 8.

7. Confirmación datos particionado:



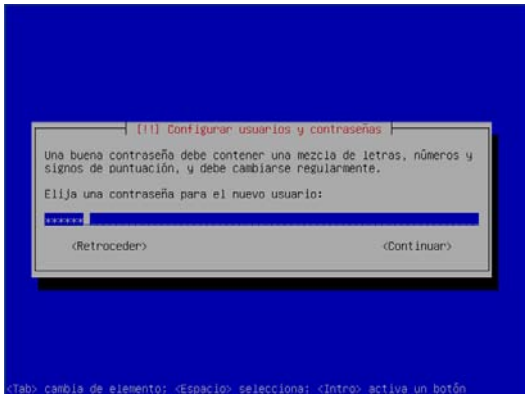
DTI. Fig 6.

8. Creación cuenta de usuario:



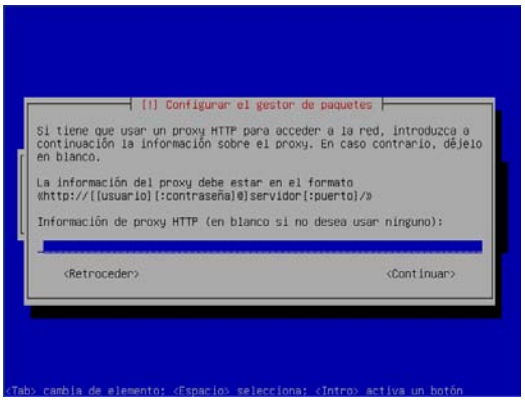
DTI. Fig 9.

9. Asignación contraseña a nuevo usuario:



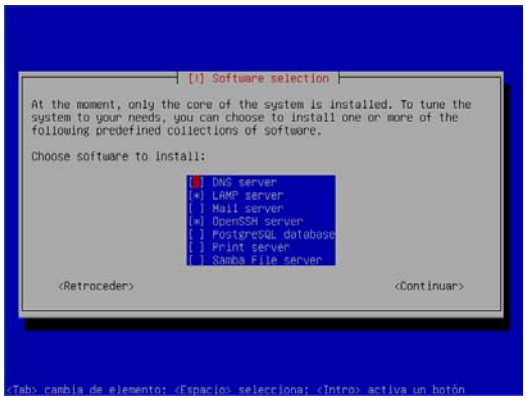
DTI. Fig 7.

10. Configuración proxy para salida Internet:



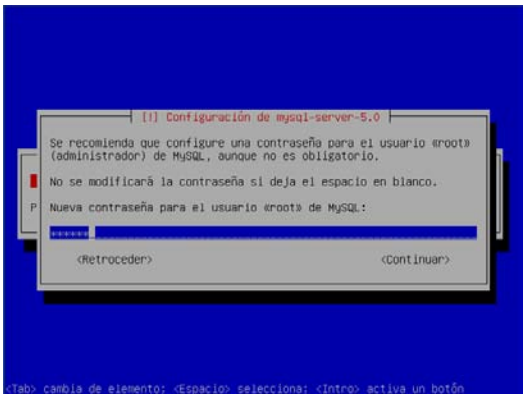
DTI. Fig 10.

11. Paquetes software a instalar: (Importante para adecuar instalación a Openfire, instalar paquetes LAMP y OpenSSH).



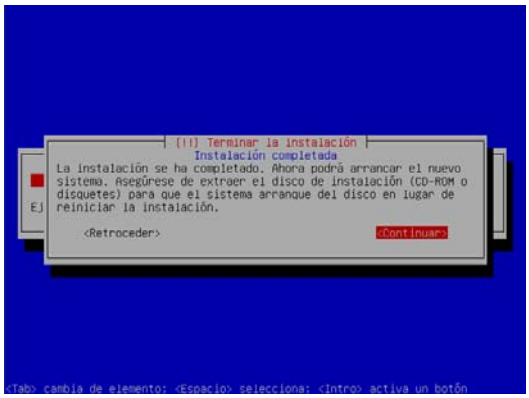
DTI. Fig 11.

12. Asignación de contraseña al usuario root de MySQL:



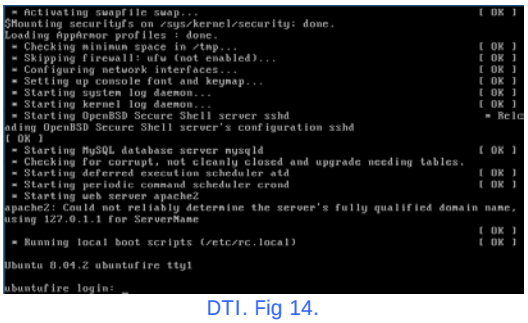
DTI. Fig 13.

13. Fin de la instalación:



DTI. Fig 12.

14. Arranque del nuevo servidor:



DTI. Fig 14.

DT.I.02. Instalación Openfire 3.6.4:

La instalación de Openfire sobre una plataforma Linux esta descrita por los siguientes pasos:

1. INSTALACIÓN MÁQUINA VIRTUAL JAVA:

Entraremos como root y nos descargamos la última versión de JRE:

```
root@imetcg:/opt# wget http://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/VerifyItem-Start/jre-6u17-linux-i586.bin?BundledLineItemUUID=35FIBe.mWx4AAAEkcXdZ7fNU&OrderID=1n9IBe.mGz4AAAEkX3dZ7fNU&ProductID=z8dIBe.n6MQAAAEkG9pUGm0.&FileName=/jre-6u17-linux-i586.bin
```

Verificaremos que se haya descargado el binario:

```
root@imetcg:/opt# ls
jre-6u17-linux-i586.bin?AuthParam=1258213045_fc8359a88269eace333442eb6c50b72d&TicketId=nod3BV0SR35+n+shnE6UVJ2adQ==&GroupName=CDS&FilePath=%2FESD6%2FJSCDL%2Fjdk%2F6u17-b04%2Fjre-6u17-linux-i586.bin&File=jre-6u17-linux-i586.bin
```

Damos permisos de ejecución:

```
root@imetcg:/opt# chmod +x jre-6u17-linux-
i586.bin\?AuthParam\=1258213045_fc8359a88269eace333442eb6c50b72d\&TicketId\=nod3BV0SR35+
n+shnE6UVJ2adQ\=\=\&GroupName\=CDS\&FilePath\=%2FESD6%2FJSCDL%2Fjdk%2F6u17-b04%2Fjre-
6u17-linux-i586.bin\&File\=jre-6u17-linux-i586.bin
```

Iniciamos la instalación de JRE:

```
root@imetcg:/opt# ./jre-6u17-linux-
i586.bin\?AuthParam\=1258213045_fc8359a88269eace333442eb6c50b72d\&TicketId\=nod3BV0SR35+
n+shnE6UVJ2adQ\=\=\&GroupName\=CDS\&FilePath\=%2FESD6%2FJSCDL%2Fjdk%2F6u17-b04%2Fjre-
6u17-linux-i586.bin\&File\=jre-6u17-linux-i586.bin
```

Verificamos la instalación:

```
root@imetcg:/opt# ls
jre1.6.0_17
jre-6u17-linux-
i586.bin?AuthParam=1258213045_fc8359a88269eace333442eb6c50b72d&TicketId=nod3BV0SR35+n+sh
nE6UVJ2adQ==&GroupName=CDS&FilePath=%2FESD6%2FJSCDL%2Fjdk%2F6u17-b04%2Fjre-6u17-linux-
i586.bin&File=jre-6u17-linux-i586.bin
```

Borramos el binario:

```
root@imetcg:/opt# rm jre-6u17-linux-
i586.bin\?AuthParam\=1258213045_fc8359a88269eace333442eb6c50b72d\&TicketId\=nod3BV0SR35+
n+shnE6UVJ2adQ\=\=\&GroupName\=CDS\&FilePath\=%2FESD6%2FJSCDL%2Fjdk%2F6u17-b04%2Fjre-
6u17-linux-i586.bin\&File\=jre-6u17-linux-i586.bin
```

Movemos la carpeta al directorio /USR y le cambiamos el nombre:

```
root@imetcg:/opt# mv /opt/jre1.6.0_17 /usr/java
```

2. INSTALACIÓN OPENFIRE:

Desde el directorio OPT, descargamos la última versión del servidor Openfire:

```
root@imetcg:/opt# wget
http://www.igniterealtime.org/downloadServlet?filename=openfire/openfire_3_6_4.tar.gz
```

Descomprimimos el fichero descargado:

```
root@imetcg:/opt# tar -xf openfire_3_6_4.tar.gz
```

Vamos a la carpeta .BIN y ejecutamos Openfire:

```
root@imetcg:/# cd /opt/openfire/bin/
root@imetcg:/opt/openfire/bin# ./openfire start
testing JVM in /usr/java ...
Starting openfire
nohup: appending output to `nohup.out'
```

Verificamos si esta arrancado Openfire:

```
root@imetcg:/opt/openfire/bin# ./openfire status
The daemon is running.
```

Aparece el mensaje: “The daemon is running”, significa que Openfire se esta ejecutando correctamente.

3. CREACIÓN Y CONFIGURACIÓN BASE DE DATOS:

Creamos la base de datos en el MySQL que tenemos instalado en el servidor:

```
root@imetcg:/opt# mysqladmin create openfire -u root -pXXXX
```

Las XXXX son la contraseña de root de mysql.

Entramos en el cliente CLI de MySQL:

```
root@imetcg:/opt# mysql -u root -pkkdvk69!
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 5.0.51a-3ubuntu5.4 (Ubuntu)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql>
```

Creamos un usuario y le otorgamos todos los permisos para administrar la base de datos de Openfire:

```
mysql> CREATE USER josvilla IDENTIFIED BY 'XXXX';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER josvilla@localhost IDENTIFIED BY 'd6agaz';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON openfire.* TO josvilla IDENTIFIED BY 'd6agaz';
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Importamos las tablas para la base de datos de Openfire. Para esto utilizaremos unas plantillas "tipo" que nos ofrece Openfire:

```
root@imetcg:/opt# mysql openfire < /opt/openfire/resources/database/openfire_mysql.sql
-u root -pXXXX
```

Ahora desde el navegador ya podremos acceder a la web de administración de Openfire para acabarlo de configurar.

DT.I.03. Instalación cliente Sparkweb:

La instalación del cliente web de mensajería instantánea de Openfire, viene descrita por los siguientes pasos:

1. Descargamos desde la web de IgniteRealtime el cliente web de Spark. El link de donde podemos descargarlo es el siguiente:

   http://www.igniterealtime.org/downloads/download-landing.jsp?file=sparkweb/sparkweb_0_9_0.tar.gz

2. Copiamos el fichero "sparkweb_0_9_0.tar.gz" en el path del servidor de Openfire: /var/www

3. Descomprimimos el fichero "sparkweb_0_9_0.tar.gz":

   ```
   root@ubuntufire:/var/www/# tar -xf sparkweb_0_9_0.tar.gz
   ```

4. Accedemos al a carpeta sparkweb y editamos el contenido del fichero SparkWeb.html y en este bloque:

   ```
   function jive_sparkweb_getConfig()
   {
           return {
                   server: "midominio.com",
                   connectionType: "socket",
                   port: "5222",
                   autoLogin: "false"
           };
   }
   ```

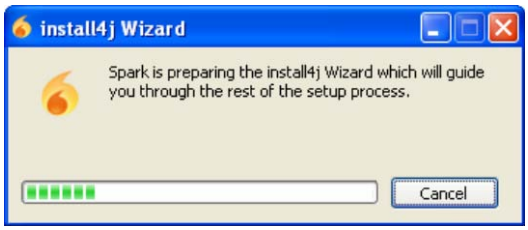
Sustituimos el nombre del dominio por nuestro hostname o dirección IP.

5. Desde un navegador web ya podremos acceder al cliente Sparkweb a través de esta dirección: http://192.168.1.14/sparkweb/SparkWeb.html

DT.I.04. Instalación cliente Spark:

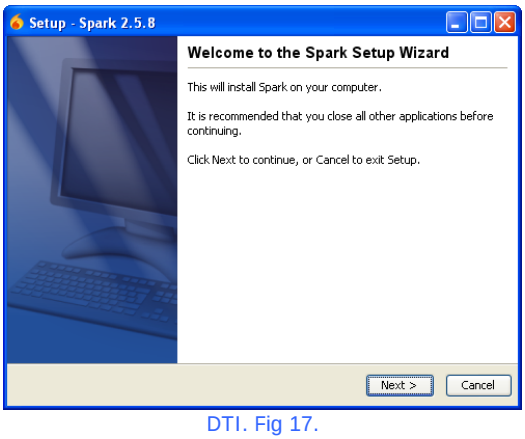
Antes de instalar el cliente de mensajería, a menos que se vaya a instalar una versión personalizada, descargaremos la última versión desde la web de IgniteRealtime: http://www.igniterealtime.org/downloads/download-landing.jsp?file=spark/spark_2_5_8.exe

1. Ejecutamos el paquete de instalación:



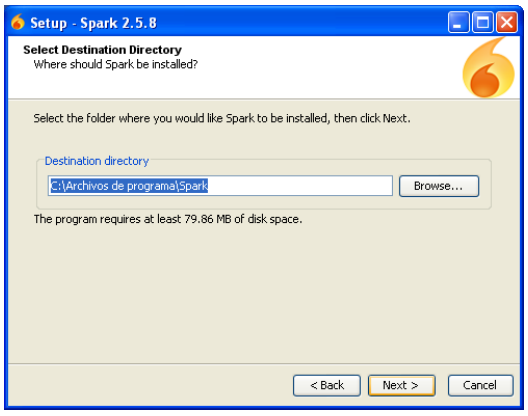
DTI. Fig 15.

2. Pulsamos sobre el botón de Next:



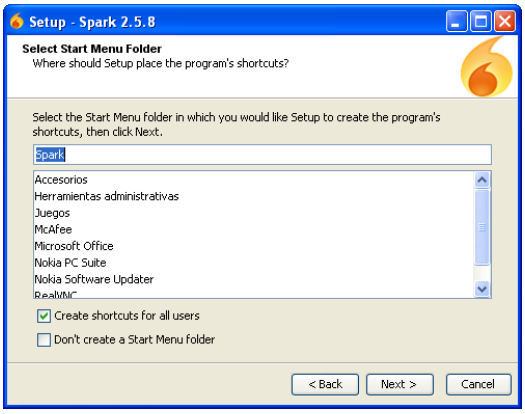
DTI. Fig 17.

3. Seleccionamos el directorio de instalación y pulsamos Next:



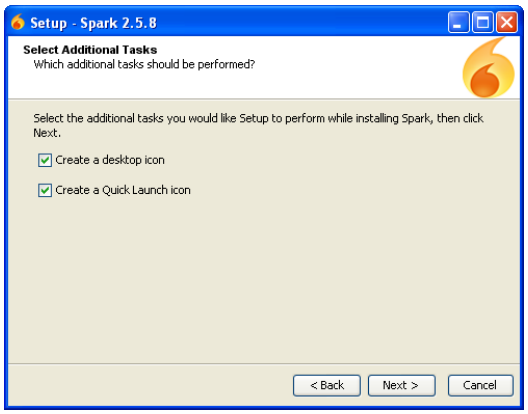
DTI. Fig 16.

4. Seleccionamos donde se van a crear los accesos del programa y pulsamos Next:



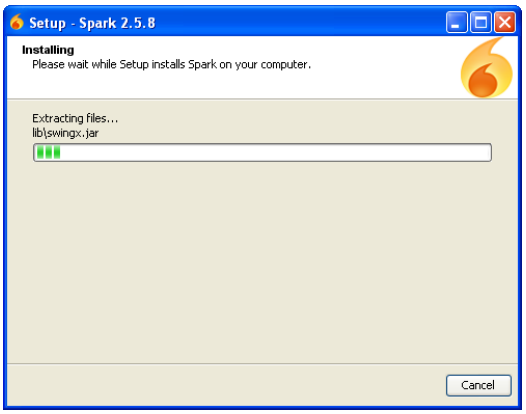
DTI. Fig 18.

5. Seleccionamos si se van a crear accesos directos y pulstamos Next:



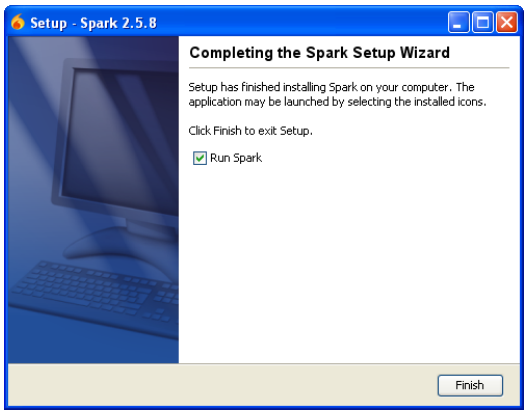
DTI. Fig 19.

6. Comienza la instalación de Spark:



DTI. Fig 21.

7. Finaliza la instalación y pulsamos Finísh sin desmarcar la opción “Run Spark”:



DTI. Fig 20.

8. Se abre el cliente Spark por primera vez, tras introducir los credenciales podremos conectarnos a Openfire:

DTI. Fig 22.

DT.I.05. Configuración pasarela XMPP:

La parametrización del módulo XMPP y el tratamiento de los mensajes IM en el servidor de Openser, se hace sobre el fichero de configuración "Openser.cfg", ubicado en la ruta: /usr/local/etc/openser

A continuación se resumen los componentes del módulo XMPP a configurar, así como una pequeña descripción de los mismos:

backend:

Permite configurar el modo de trabajo del módulo XMPP. Las opciones de configuración son:

- Component.
- Server.

Si se configura como Component, el módulo XMPP de Openser se registrará como componente externo en una plataforma jabber externa.
Si se configura como Server, será el mismo módulo XMPP de Openser quien se encargará de la gestión del tráfico XMPP recibido de otra plataforma jabber.

domain_separator:

Además del carácter especial @, las direcciones destino de los usuarios Jabber y SIP necesitarán de un carácter extra como separador entre dominios.

gateway_domain:

Este parámetro configura el dominio donde esta el servidor de Openser. Este dominio debe estar accesible tanto desde los servidores SIP y Jabber como desde los clientes de IM.

xmpp_domain:

Este parámetro configura el dominio del servidor xmpp, o sea, el dominio de la plataforma Openfire.

Dependiendo de la instalación, las plataformas Openser y Openfire pueden estar en el mismo dominio, por lo tanto los parámetros xmpp_domain y gateway_domain harían referencia al mismo dominio.

xmpp_host:

Mediante este parámetro configuramos la dirección IP o hostname del servidor jabber, si se ha configurado el backend como component o la dirección IP por la que deberán entrar las peticiones XMPP de plataformas jabber externas, si el backend se ha configurado como server.

Por lo tanto, la parametrización para establecer una pasarela XMPP en nuestro entorno de trabajo virtual es la siguiente:

```
modparam("xmpp", "backend", "component")
modparam("xmpp", "domain_separator", "*")
modparam("xmpp", "gateway_domain", "192.168.1.9")
modparam("xmpp", "xmpp_domain", "192.168.1.9.192.168.1.14")
modparam("xmpp", "xmpp_host", "192.168.1.14")
```

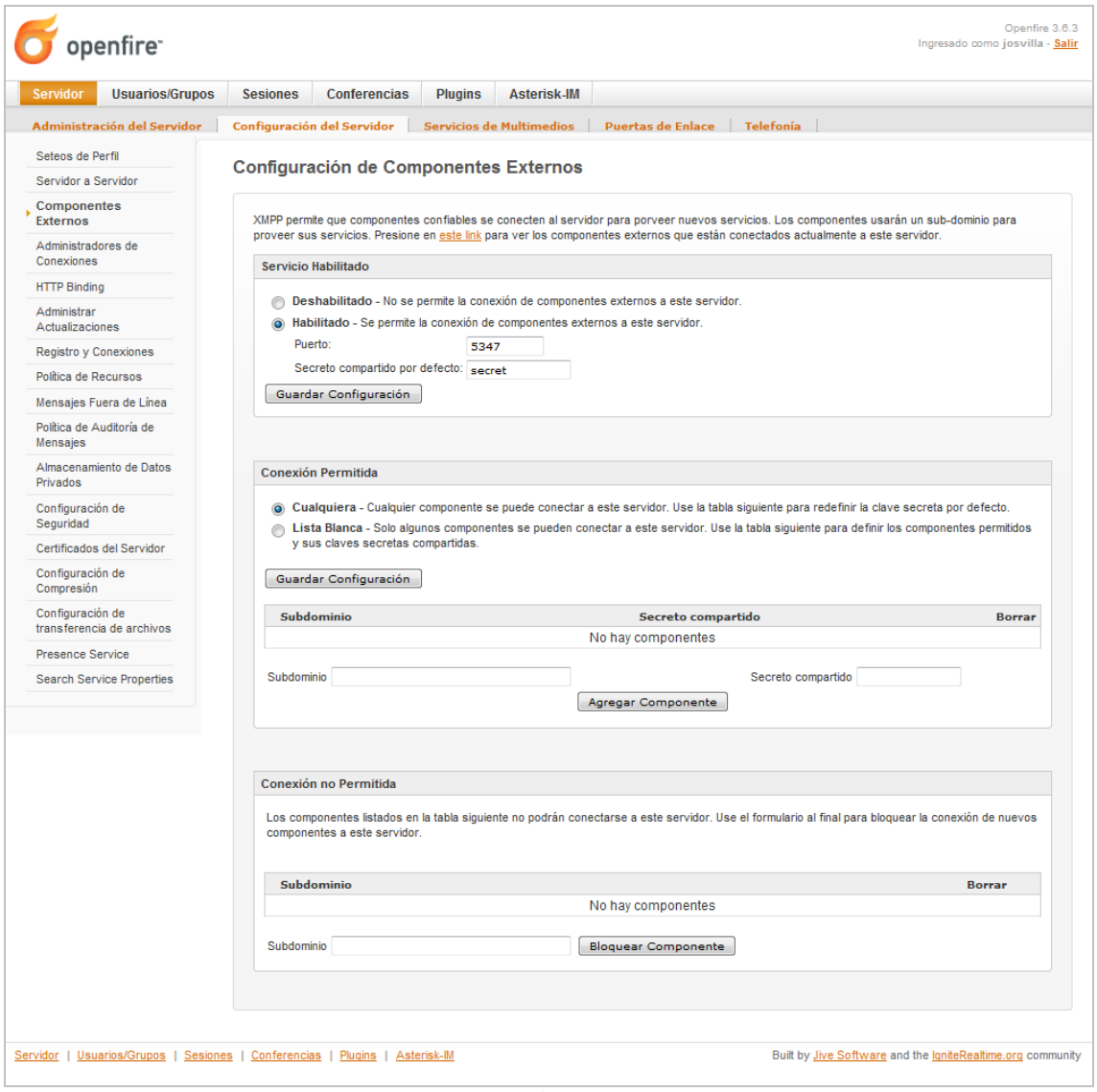
El tratamiento configurado para los mensajes de IM-XMPP es el siguiente:

```
# XMPP-MESSAGE processing
if (is_method("MESSAGE") && avp_check("$rd", "eq/s:gw"))
{
          route(3);
}
```

```
#XMPP MESSAGE route
route[3]
{
        if (!t_newtran())
        {
                sl_reply_error();
                exit;
        }
        if (xmpp_send_message())
        {
                t_reply("200", "Accepted");
        } else {
                t_reply("404", "Not found");
        }
        exit;
}
```

En el documento técnico de instalación anexo, “DT.I.07. Configuración básica Openser”, se muestra el contenido completo del fichero configuración de la plataforma Openser del laboratorio virtual.

En cuanto al servidor de Openfire, deberá tener habilitadas las conexiones de componentes externos y deberá de estar permitida la conexión del servidor de Openser. Esto se habilita a través de la web de administración de Openfire, accediendo a la pestaña de servidor, configuración del servidor y haciendo clic sobre componentes externos:



DTI. Fig 23. Configuración componentes externos.

Para validar que el servidor Openser se ha registrado como componente XMPP externo en la plataforma de Openfire, basta con acceder a través de la web de administración de Openfire a la pestaña Sesiones, Sesiones de componentes. Si se ha registrado correctamente aparecerá el componente registrado tal y como se muestra a continuación:

DTI. Fig 24. Sesiones componentes externos.

Con estas parametrizaciones queda establecida la pasarela XMPP entre Openser y Openfire.

Ahora los clientes IM-Jabber que quieran mapear a clientes de IM-SIP deberán mapearlos con la siguiente nomenclatura: <usuario_sip>*<gateway_domain>@<xmpp_domain>
Por ejemplo: 2150*192.168.1.9@192.168.1.9.192.168.1.14

Y los clientes IM-SIP que quieran mapear a clientes IM-Jabber deberán hacerlo con la siguiente nomenclatura: <usuario_sip>*<xmpp_host>@<gateway_domain>
Por ejemplo: josvilla*192.168.1.14@192.168.1.9

DT.I.06. Configuración traspaso de presencia:

La parametrización del módulo PUA_XMPP (Presence User Agent for XMPP) y el tratamiento de los mensajes de presencia en el servidor de Openser, se hace sobre el fichero de configuración "Openser.cfg", ubicado en la ruta: /usr/local/etc/openser.

Para habilitar el módulo PUA_XMPP es mandatario tener habilitados los módulos: Presence, PUA (Presence User Agent) y el módulo XMPP, descrito como se habilitaba y configuraba en el DT.I.05.

La documentación técnica de fabricante sobre cómo se implementan estos módulos se puede revisar en los siguientes repositorios:

Presence ⇨ http://www.kamailio.org/docs/modules/1.3.x/presence.html

PUA ⇨ http://www.kamailio.org/docs/modules/1.3.x/pua.html

PUA_XMPP ⇨ http://www.kamailio.org/docs/modules/1.3.x/pua_xmpp.html

A continuación se hace una descripción de las funciones de cada uno de estos módulos y de las parametrizaciones de sus componentes:

Presence:

Este módulo implementa el core de la funcionalidad de notificación de eventos SIP. Gestiona los mensajes PUBLISH y SUBSCRIBE y genera mensajes NOTIFY.

db_url:

Configurando este parámetro, el módulo de presencia queda habilitado. La configuración de este componente hace referencia a la localización de las tablas de Openser, se configura donde esta el mysql que contiene estas tablas y que base de datos usa.

server_address:

Este parametro permite configurar la dirección del servidor de presencia, sobre este servidor se enviarán los mensajes de SUBSCRIBE y PUBLISH y se recibirán los mensajes NOTIFY.

PUA:

PUA significa Presence User Agent y configurándolo queda habilitado un cliente de Presencia enviando mensajes tipo PUBLISH y SUBSCRIBE.

db_url:

De la misma forma que hace el módulo Presence, este módulo necesita conocer la localización de las tablas donde openser guarda su información. Por lo que se configura donde esta el mysql que contiene estas tablas y que base de datos usa.

db_table:

A través de este parametro se configura el nombre de la tabla donde se almacenan los mensajes PUBLISH y SUBSCRIBE.

Tras habilitar este módulo se pueden habilitar el módulo PUA_XMPP.

PUA_XMPP:

PUA_XMPP significa Presence User Agent Subscribe for XMPP. Con este modulo se establece una pasarela que traspasa presencia entre SIP y XMPP y viceversa.

Se encarga de traducir los mensajes SIP de presencia en mensajes XMPP y viceversa.

server_address:

A través de este parámetro se indica la dirección IP donde se encuentra el servidor de Presencia que se hará cargo de la traducción de mensajes SIP-XMPP-SIP.

Por lo tanto, la parametrización para establecer una pasarela de presencia en nuestro entorno de trabajo virtual es la siguiente:

```
#PARAMETRIZACIÓN MODULO PRESENCE
modparam("presence", "db_url", "mysql://openser:openserrw@localhost/openser")
modparam("presence", "server_address", "sip:192.168.1.9")

#PARAMETRIZACIÓN MODULO PUA
modparam("pua", "db_url", "mysql://openser:openserrw@localhost/openser")
modparam("pua", "db_table", "pua")

#PARAMETRIZACIÓN MODULO PUA_XMPP
modparam("pua_xmpp", "server_address", "sip:192.168.1.9")
```

El tratamiento configurado para los mensajes la gestión de los mensajes PULBISH, SUBSCRIBE Y NOTIFY es el siguiente:

```
# XMPP-PRESENCE processing
if (is_method("PUBLISH|SUBSCRIBE|NOTIFY") && uri=~"sip:.+@gw")
{
         route(2);
}

if (is_method("PUBLISH|SUBSCRIBE") && uri==myself)
{
        route(2);
}
```

```
route[2]
{
        if (!t_newtran())
        {
                sl_reply_error();
                exit;
        };

        if( is_method("NOTIFY") )
        {
                if(  uri=~"sip:.+@gw"  ||  (  $rd  ==  "192.168.1.9"  &&  $hdr(Event)==
"presence.winfo" ) ) {
                        pua_xmpp_notify();
                        t_reply("200", "OK");
                }
                else
                        t_release();
        }
        else
        if(is_method("PUBLISH"))
        {
                handle_publish();
                t_release();
        }
        else
        if( is_method("SUBSCRIBE"))
        {
                handle_subscribe();
                if( uri=~"sip:.+@gw" && $hdr(Event)== "presence" )
                {
                        pua_xmpp_req_winfo("$ruri", "$hdr(Expires)");
                }
                t_release();
        }
        exit;
}
```

En el documento técnico de instalación anexo, “DT.I.07. Configuración básica Openser”, se muestra el contenido completo del fichero configuración de la plataforma Openser del laboratorio virtual.

En cuanto al servidor Openfire no es necesaria ninguna parametrización más que la indicada para establecer la pasarela XMPP.

Los usuarios SIP y Jabber deberán mapearse entre ellos siguiendo la nomenclatura indicada en el DT.I.05.

DT.I.07. Configuración básica Openser:

El fichero de configuración de Openser, se encuentra en la siguiente ruta del servidor de Openser:

/usr/local/etc/openser/openser.cfg

Para el entorno de laboratorio virtual sobre el que se ha trabajado, se ha instalado un Proxy SIP con Openser y con una configuración básica, pero que implementa la pasarela XMPP para el intercambio de mensajes instantáneos entre el mundo Jabber y el mundo SIP y también implementa el traspaso de Presencia entre ambos mundos usando la pasarela XMPP.

A continuación se muestra el printado del fichero de configuración del Openser implementado en nuestra plataforma de desarrollo.

```
# $Id: openser.cfg 3284 2007-12-06 18:56:59Z bogdan_iancu $
#
# OpenSER basic configuration script
#


####### Global Parameters #########

debug=3
log_stderror=no
log_facility=LOG_LOCAL0

fork=yes
children=4

/* uncomment the following lines to enable debugging */
#debug=6
#fork=no
#log_stderror=yes

/*TCP (default on) */
disable_tcp=no


port=5060
#sip server alias
alias="ubuntukamailio"
alias="192.168.1.9"


####### Modules Section ########
#set module path
mpath="/usr/local/lib/openser/modules/"
loadmodule "mysql.so"
loadmodule "sl.so"
loadmodule "tm.so"
loadmodule "rr.so"
loadmodule "maxfwd.so"
loadmodule "usrloc.so"
loadmodule "registrar.so"
loadmodule "textops.so"
loadmodule "mi_fifo.so"
loadmodule "uri_db.so"
loadmodule "uri.so"
loadmodule "xlog.so"
loadmodule "acc.so"
loadmodule "mi_xmlrpc.so"
loadmodule "domain.so"
loadmodule "presence.so"
loadmodule "presence_xml.so"
loadmodule "avpops.so"
loadmodule "xmpp.so"
loadmodule "pua.so"
loadmodule "pua_xmpp.so"



# ----------------- setting module-specific parameters ---------------

# ----- mi_fifo params -----
modparam("mi_fifo", "fifo_name", "/tmp/openser_fifo")


# ----- rr params -----
modparam("rr", "enable_full_lr", 1)
modparam("rr", "append_fromtag", 0)


# ----- rr params -----
modparam("registrar", "method_filtering", 1)

# ----- acc params -----
/* what sepcial events should be accounted ? */
```

```
modparam("acc", "early_media", 1)
modparam("acc", "report_ack", 1)
modparam("acc", "report_cancels", 1)
modparam("acc", "detect_direction", 0)

/* account triggers (flags) */
modparam("acc", "failed_transaction_flag", 3)
modparam("acc", "log_flag", 1)
modparam("acc", "log_missed_flag", 2)

/*DB accounting also */
modparam("acc", "db_flag", 1)
modparam("acc", "db_missed_flag", 2)


# ----- usrloc params -----

/* enable DB persistency for location entries */

modparam("usrloc", "db_mode",   2)
modparam("usrloc", "db_url", "mysql://openser:openserrw@localhost/openser")


# ----- domain params -----
/* multi-domain detection support */
modparam("domain", "db_url","mysql://openser:openserrw@localhost/openser")
modparam("domain", "db_mode", 1)   # Use caching


# ----- multi-module params -----
/* multi-domain support in the modules (dafault off) */
modparam("alias_db|auth_db|usrloc|uri_db", "use_domain", 1)


# ----- presence and pua params -----
modparam("presence|presence_xml","db_url",
"mysql://openser:openserrw@localhost/openser")
modparam("pua", "db_url", "mysql://openser:openserrw@localhost/openser")
modparam("pua", "db_table", "pua")
modparam("presence", "max_expires", 3600)
modparam("presence", "server_address", "sip:192.168.1.9")
modparam("presence", "fallback2db", 0)

# ----- new pua_xmpp param: ip address of the server -----
modparam("pua_xmpp", "server_address", "sip:192.168.1.9")

# ----- XMPP params -----
modparam("xmpp", "backend", "component")
modparam("xmpp", "domain_separator", "*")
modparam("xmpp", "gateway_domain", "gw")
modparam("xmpp", "xmpp_domain", "192.168.1.9.192.168.1.14")
modparam("xmpp", "xmpp_host", "192.168.1.14")

# ----- xmlrpc params -----
modparam("mi_xmlrpc", "port", 8008)
modparam("mi_xmlrpc", "log_file", "/var/log/abyss.log")
modparam("mi_xmlrpc", "reply_option", 0)
modparam("mi_xmlrpc", "buffer_size", 8192)

# ----- Presence_xml treatment -----
modparam("presence_xml", "force_active", 1)
modparam("presence_xml", "xcap_table", "xcap")
modparam("presence_xml", "integrated_xcap_server", 1)
modparam("presence_xml", "xcap_server", "http://192.168.1.9:8000")

######## Routing Logic ########


# main request routing logic

route
{

        if (is_method("INVITE"))
        {
                sl_send_reply("200","OK");
                exit;
```

```
        }

        if (!mf_process_maxfwd_header("10")) {
                sl_send_reply("483","Too Many Hops");
                exit;
        }

        if (has_totag()) {
                if (loose_route()) {
                        if (is_method("BYE")) {
                                setflag(1); # do accouting ...
                                setflag(3); # ... even if the transaction fails
                        }
                        route(1);
                } else {
                        /* enable presence */
                        if ( is_method("SUBSCRIBE|NOTIFY") )
                       {
                                route(2);
                                exit;
                        }
                        if ( is_method("ACK") ) {
                                if ( t_check_trans() )
                             {
                                        t_relay();
                                        exit;
                                }
                             else
                             {
                                        exit;
                                }
                        }
                        sl_send_reply("404","Not here");
                }
                exit;
        }


        #initial requests


        # XMPP-PRESENCE processing
        if (is_method("PUBLISH|SUBSCRIBE|NOTIFY") && uri=~"sip:.+@gw")
       {
                route(2);
        }

        if (is_method("PUBLISH|SUBSCRIBE") && uri==myself)
       {
                route(2);
        }


        # XMPP-MESSAGE processing
        if (is_method("MESSAGE") && avp_check("$rd", "eq/s:gw"))
       {
                route(3);
        }

        # CANCEL processing
        if (is_method("CANCEL"))
        {
                if (t_check_trans())
                        t_relay();
                exit;
        }

        t_check_trans();

        # record routing
        if (!is_method("REGISTER|MESSAGE"))
                record_route();

        # account only INVITEs
        if (is_method("INVITE")) {
                setflag(1); # do accouting
        }
```

```
        if (!uri==myself)
        {
                append_hf("P-hint: outbound\r\n");
                route(1);
        }

        # requests for my domain


        if (is_method("PUBLISH"))
        {
                sl_send_reply("503", "Service Unavailable");
                exit;
        }

        if (is_method("REGISTER"))
        {
                if (!save("location"))
                        sl_reply_error();
                exit;
        }

        if ($rU==NULL)
        {
                # request with no Username in RURI
                sl_send_reply("484","Address Incomplete");
                exit;
        }

        if (!lookup("location"))
        {
                switch ($retcode)
                {
                        case -1:
                        case -3:
                                t_newtran();
                                t_reply("404", "Not Found");
                                exit;
                        case -2:
                                sl_send_reply("405", "Method Not Allowed");
                                exit;
                }
        }

        # when routing via usrloc, log the missed calls also
        setflag(2);

        route(1);
}


route[1]
{
        # for INVITEs enable some additional helper routes
        if (is_method("INVITE"))
        {
                t_on_branch("2");
                t_on_reply("2");
                t_on_failure("1");
        }

        if (!t_relay())
        {
                sl_reply_error();
        };
        exit;
}


# Presence route
route[2]
{
        if (!t_newtran())
        {
                sl_reply_error();
                exit;
        };
```

```
        if( is_method("NOTIFY") )
        {
                if(  uri=~"sip:.+@gw"  ||  (  $rd  ==  "192.168.1.9"  &&  $hdr(Event)==
"presence.winfo" ) )
               {
                        pua_xmpp_notify();
                        t_reply("200", "OK");
                }
                else
                        t_release();
        }
        else
        if(is_method("PUBLISH"))
        {
                handle_publish();
                t_release();
        }
        else
        if( is_method("SUBSCRIBE"))
        {
                handle_subscribe();
                if( uri=~"sip:.+@gw" && $hdr(Event)== "presence" )
                {
                        pua_xmpp_req_winfo("$ruri", "$hdr(Expires)");
                }
                t_release();
        }
        exit;
}


#XMPP MESSAGE route
route[3]
{
        if (!t_newtran())
        {
                sl_reply_error();
                exit;
        }
        if (xmpp_send_message())
        {
                t_reply("200", "Accepted");
        }
        else
        {
                t_reply("404", "Not found");
        }
        exit;
}


branch_route[2]
{
        xlog("new branch at $ru\n");
}


onreply_route[2]
{
        xlog("incoming reply\n");
}


failure_route[1]
{
        if (t_was_cancelled())
        {
                exit;
        }
}
```

DT.I.08. Integración Openfire con OpenLDAP:

Para dar de alta en openfire los datos del servidor de LDAP, en caso de tratarse de una instalación ya en producción, debemos entrar en el modo de configuración, para ello seguiremos lo siguientes pasos:

1. Entrar en el directorio de configuración de openfire:
   root@ubuntufire:/opt/openfire/conf#

2. Editar el fichero openfire.xml.
3. Modificar la línea:
   <setup>**true**</setup> ⇨ <setup>**false**</setup>

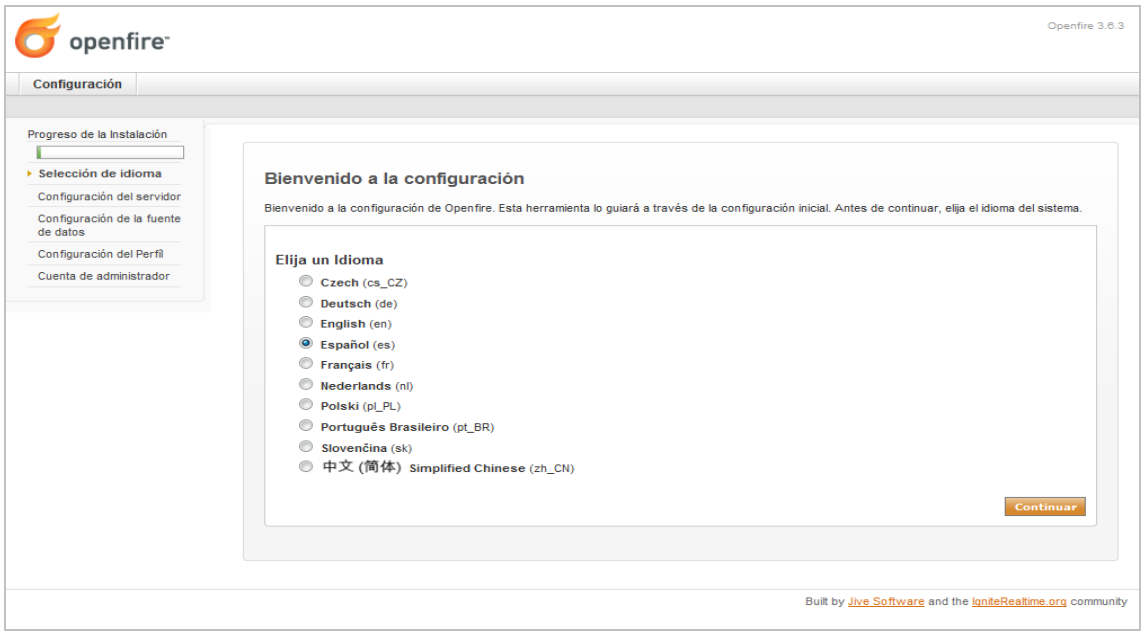
4. Entrar de nuevo en la página web de administración de openfire.

De esta forma, al entrar nos encontraremos con esta página:

DTI. Fig 25. Pantallas integración con OpenLDAP.

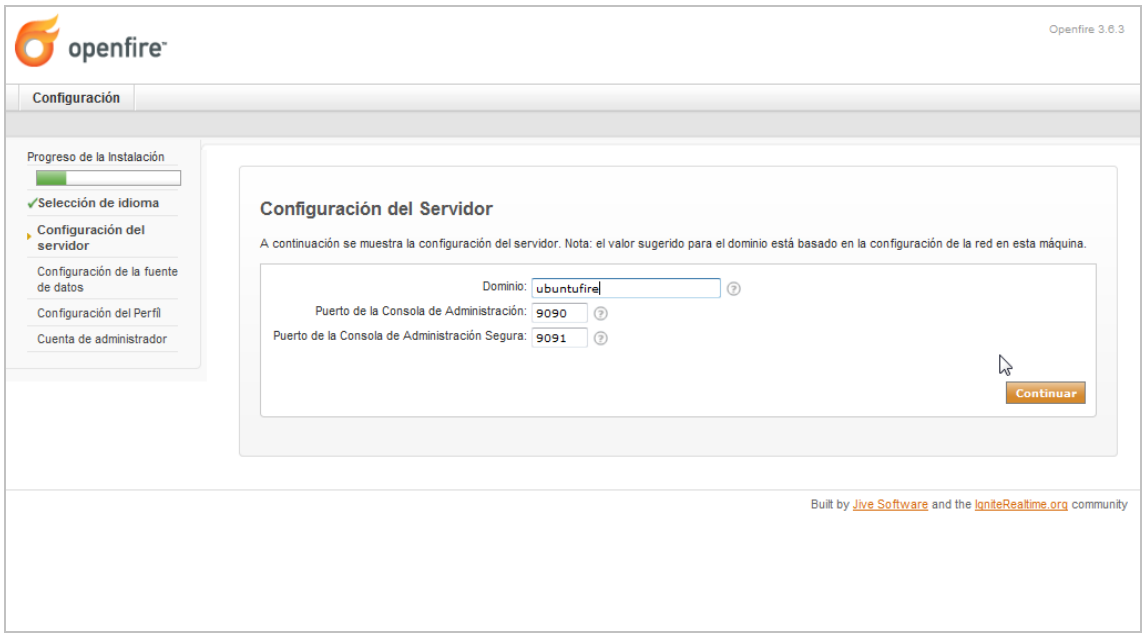
A partir de estos preliminares, se deben seguir los pasos descritos a continuación para completar la integración con un servidor de OpenLDAP.

1. Selección de idioma:



DTI. Fig 26.

2. Configuración Servidor:

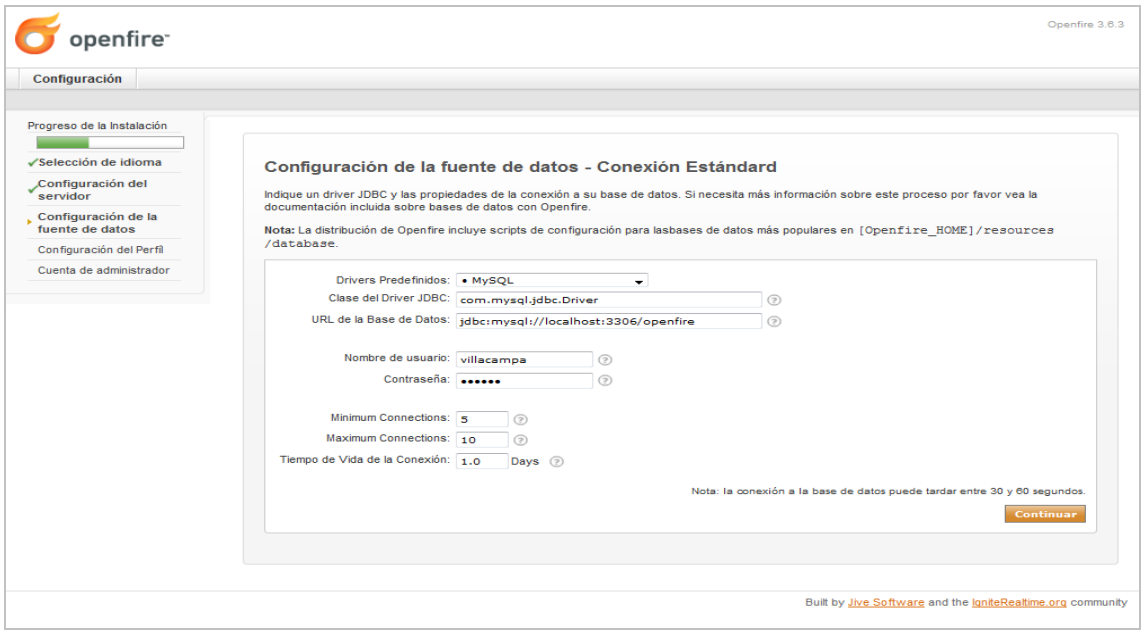


DTI. Fig 27.

3. Configuración tipo de base de datos:

DTI. Fig 28.

4. Configuración tipo base de datos y datos de conexión:

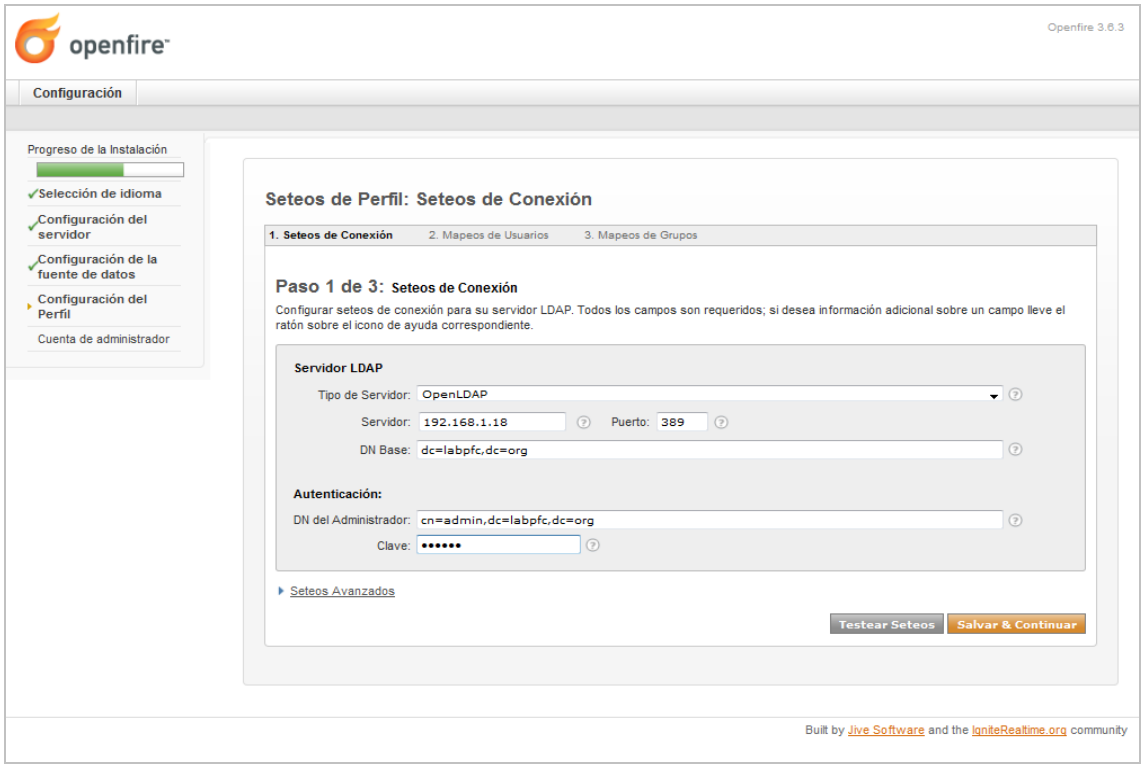


DTI. Fig 29.

5. Selección sistema de gestión de usuarios:

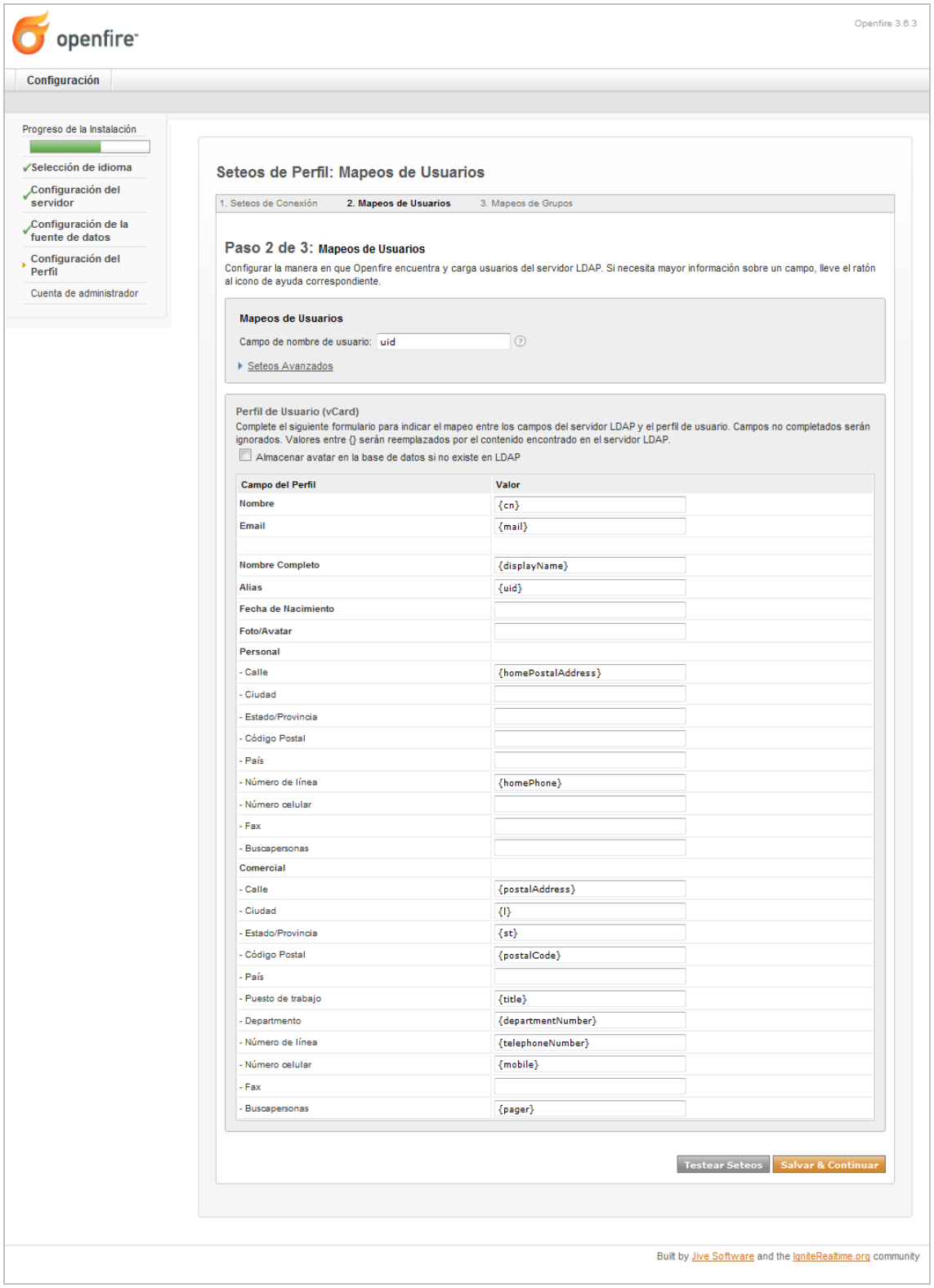
DTI. Fig 30.

6. Datos de conexión OpenLDAP:



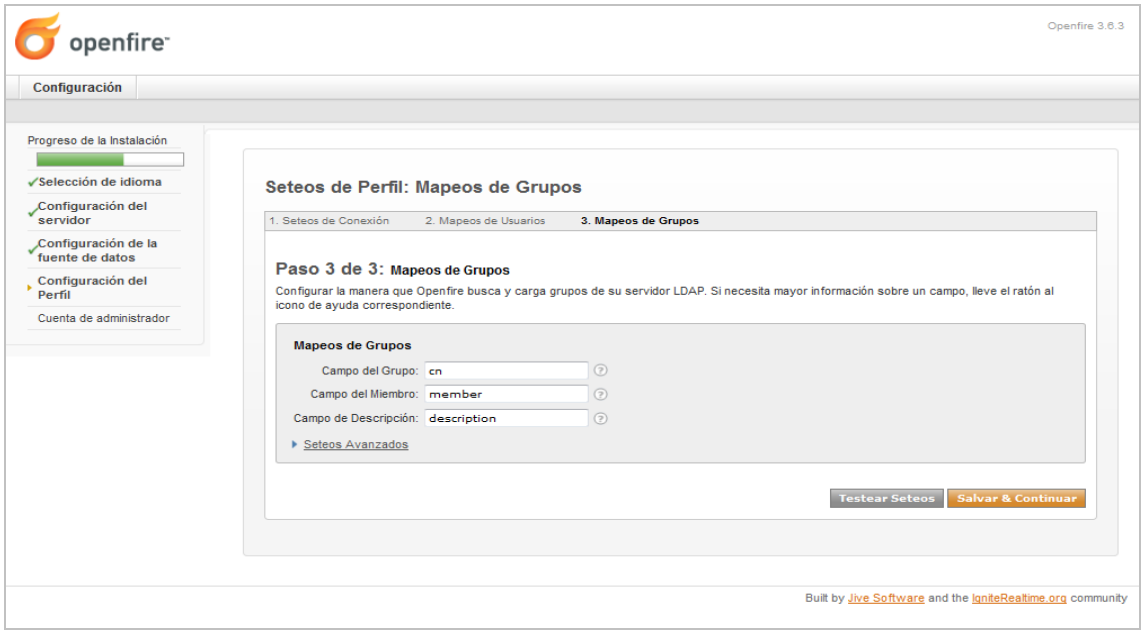
DTI. Fig 31.

7. Mapeo campos usuario OpenLDAP en Openfire:



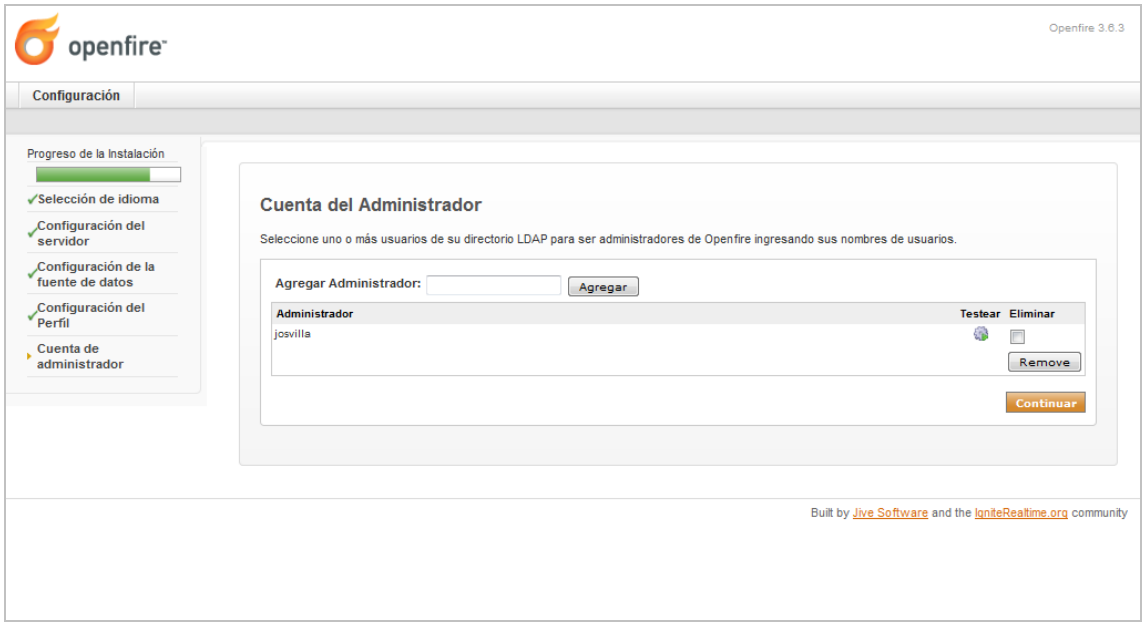
DTI. Fig 32.

8. Mapeo campos grupo OpenLDAP en Openfire:



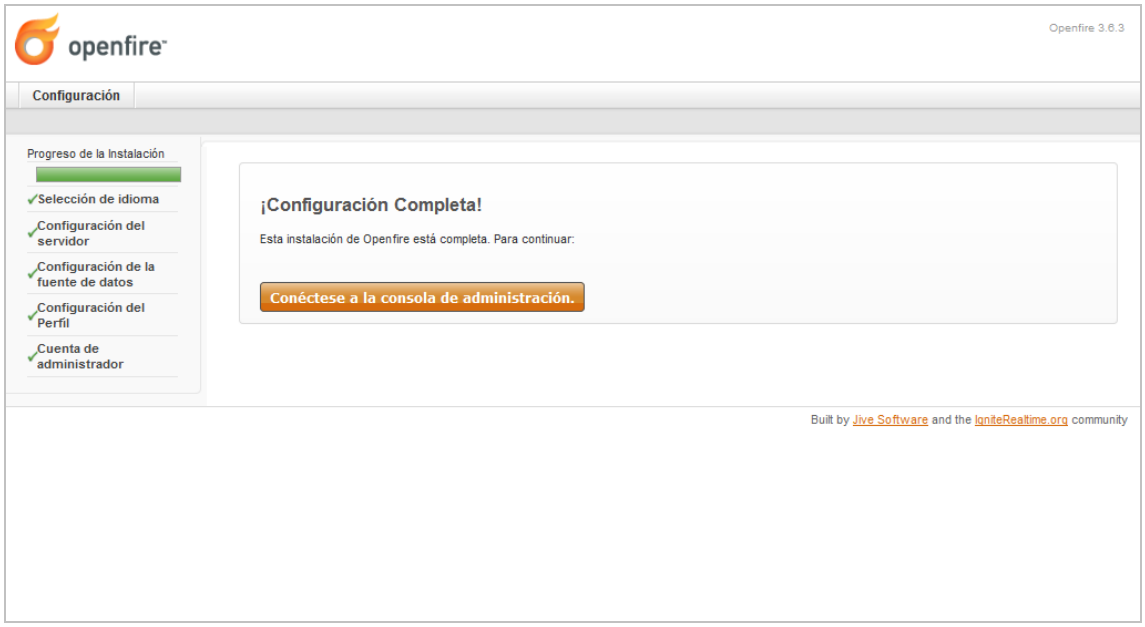
DTI. Fig 33.

9. Mapeo usuario administrador:



DTI. Fig 34.

10. Fin de la configuración:



DTI. Fig 35.

Importante dar de alta un usuario administrador y guardar sus datos, sino no podrá validarse ningún administrador en la consola de administración de openfire.

DT.I.09. Integración Openfire con Directorio Activo:

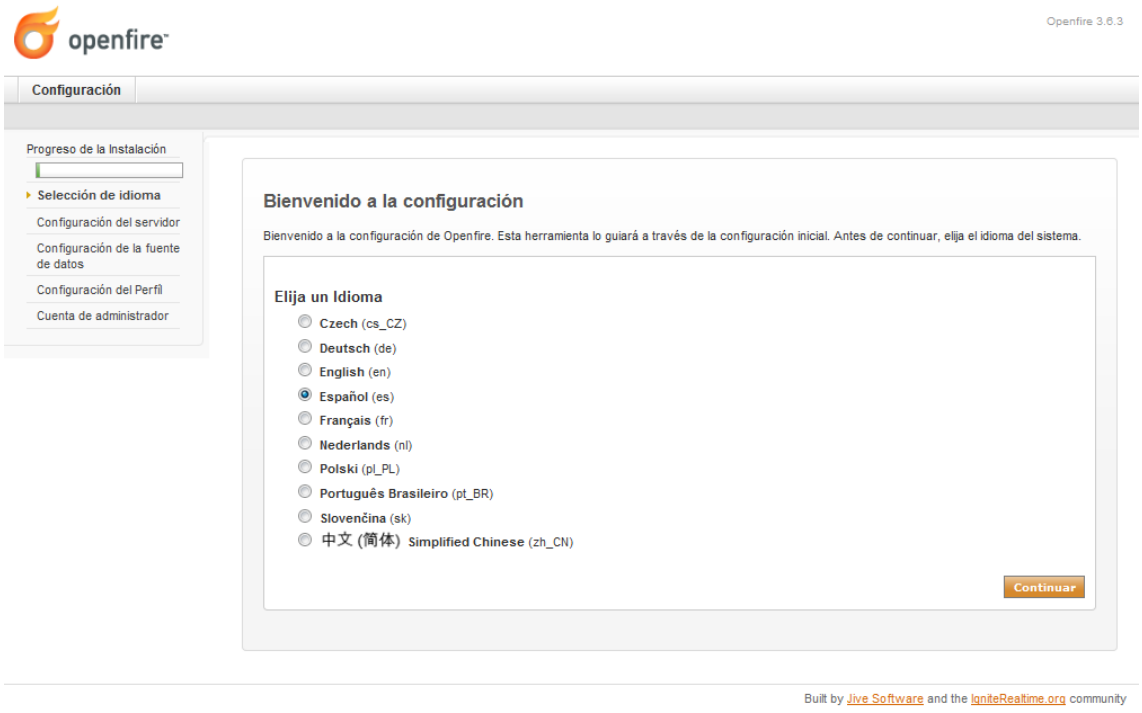
Para dar de alta en openfire los datos del servidor de LDAP, en caso de tratarse de una instalación ya en producción, debemos entrar en el modo de configuración, para ello seguiremos lo siguientes pasos:

7. Entrar en el directorio de configuración de openfire:
   root@ubuntufire:/opt/openfire/conf#

8. Editar el fichero openfire.xml.
9. Modificar la línea:
   <setup>**true**</setup> ⇨ <setup>**false**</setup>

10. Entrar de nuevo en la página web de administración de openfire.

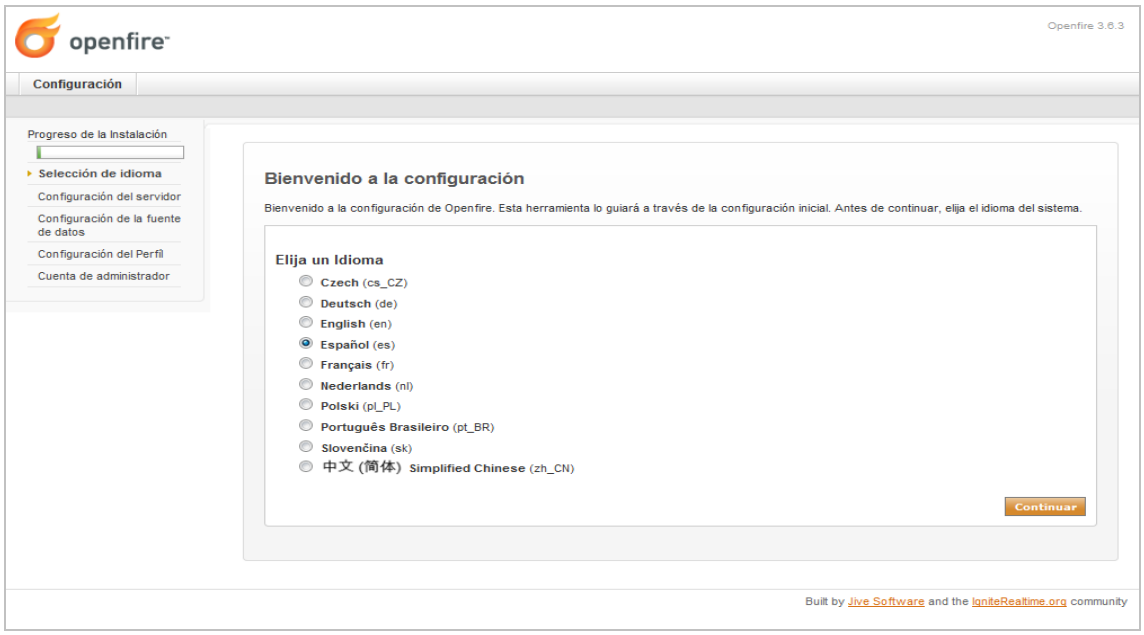
De esta forma, al entrar nos encontraremos con esta página:



DTI. Fig 36. Pantallas integración con Directorio Activo.

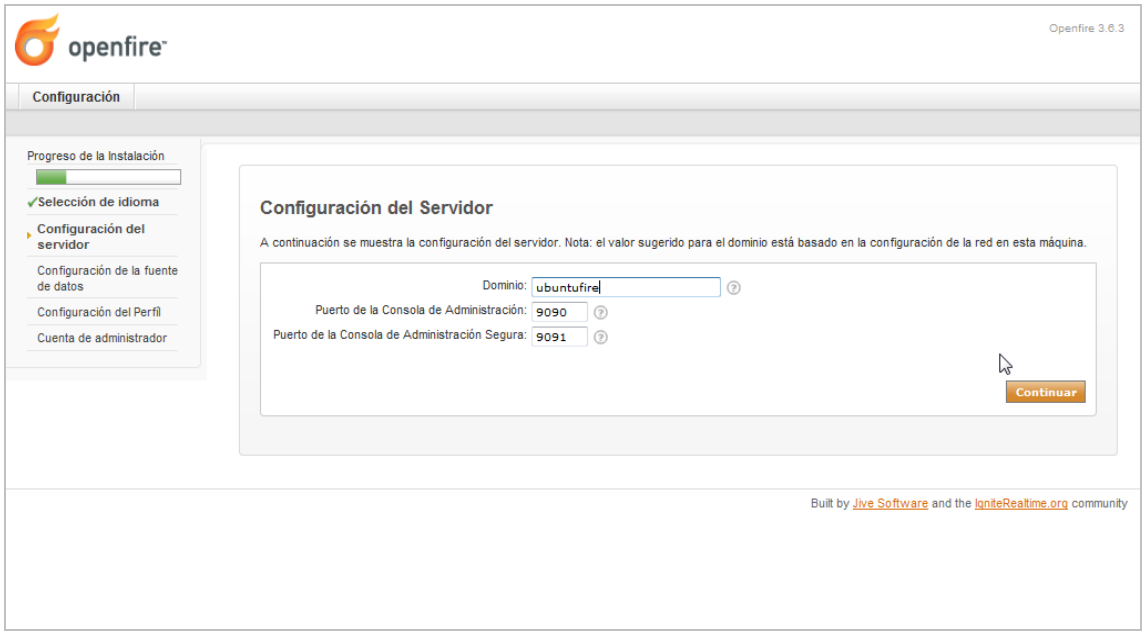
A partir de estos preliminares, se deben seguir los pasos descritos a continuación para completar la instalación integrando un servidor de Directorio Activo.

1. Selección de idioma:



DTI. Fig 37.

2. Configuración Servidor:

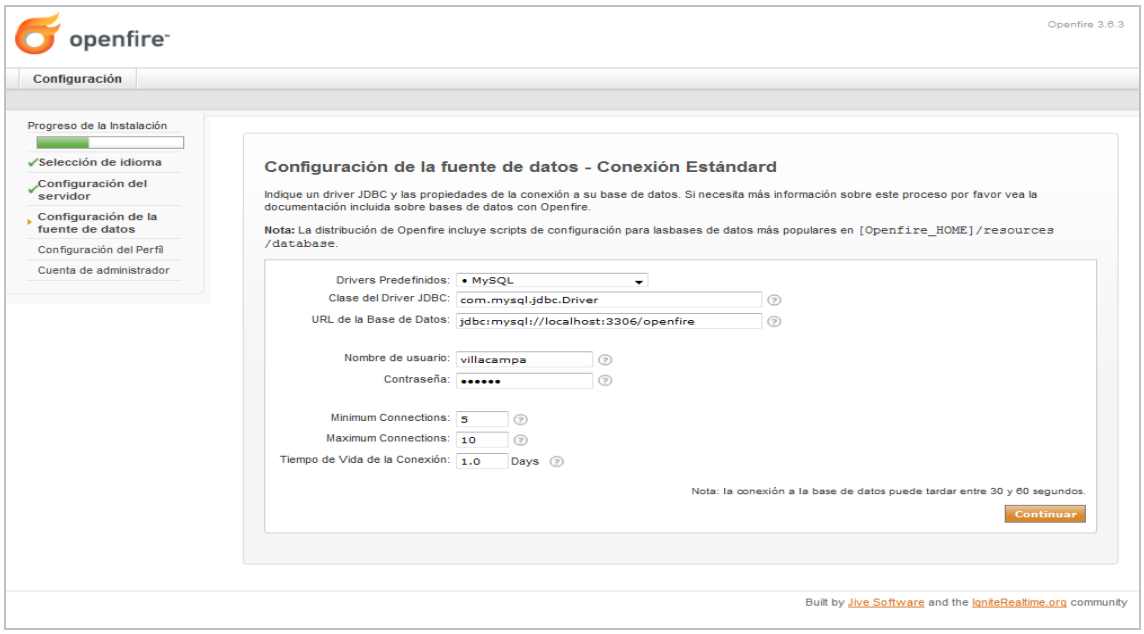


DTI. Fig 38.

3. Configuración tipo de base de datos:

DTI. Fig 39.

4. Configuración tipo base de datos y datos de conexión:

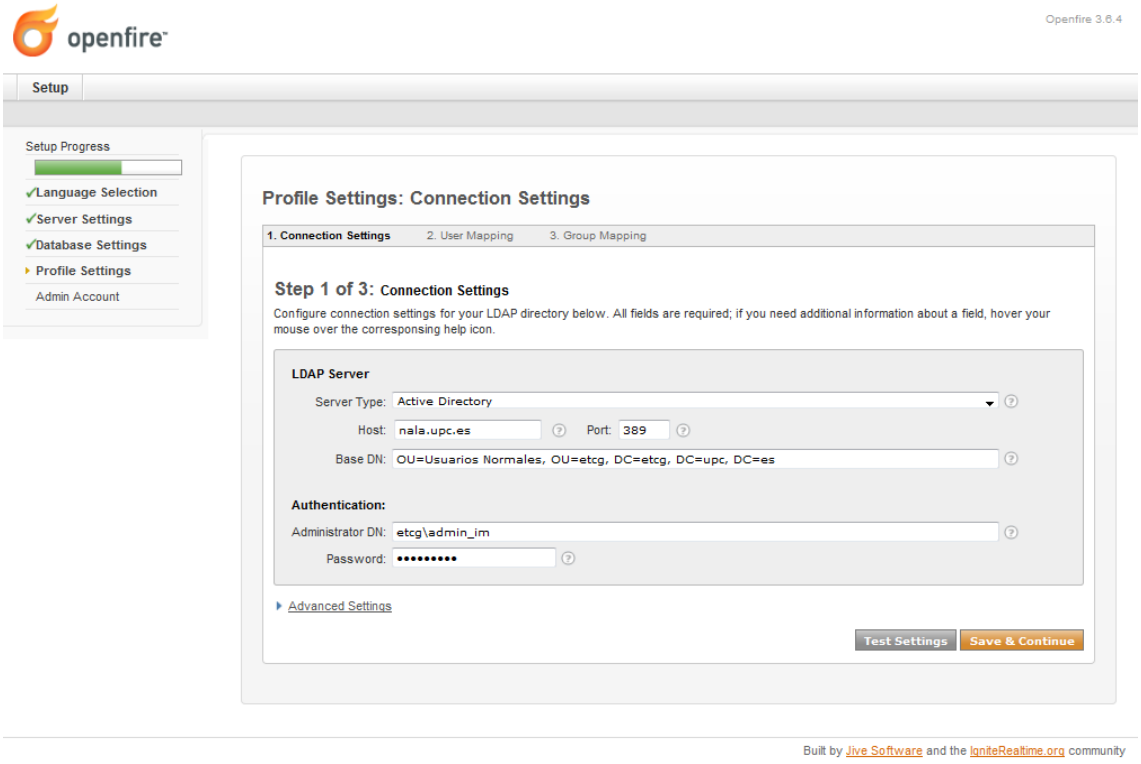


DTI. Fig 40.

5. Selección sistema de gestión de usuarios:

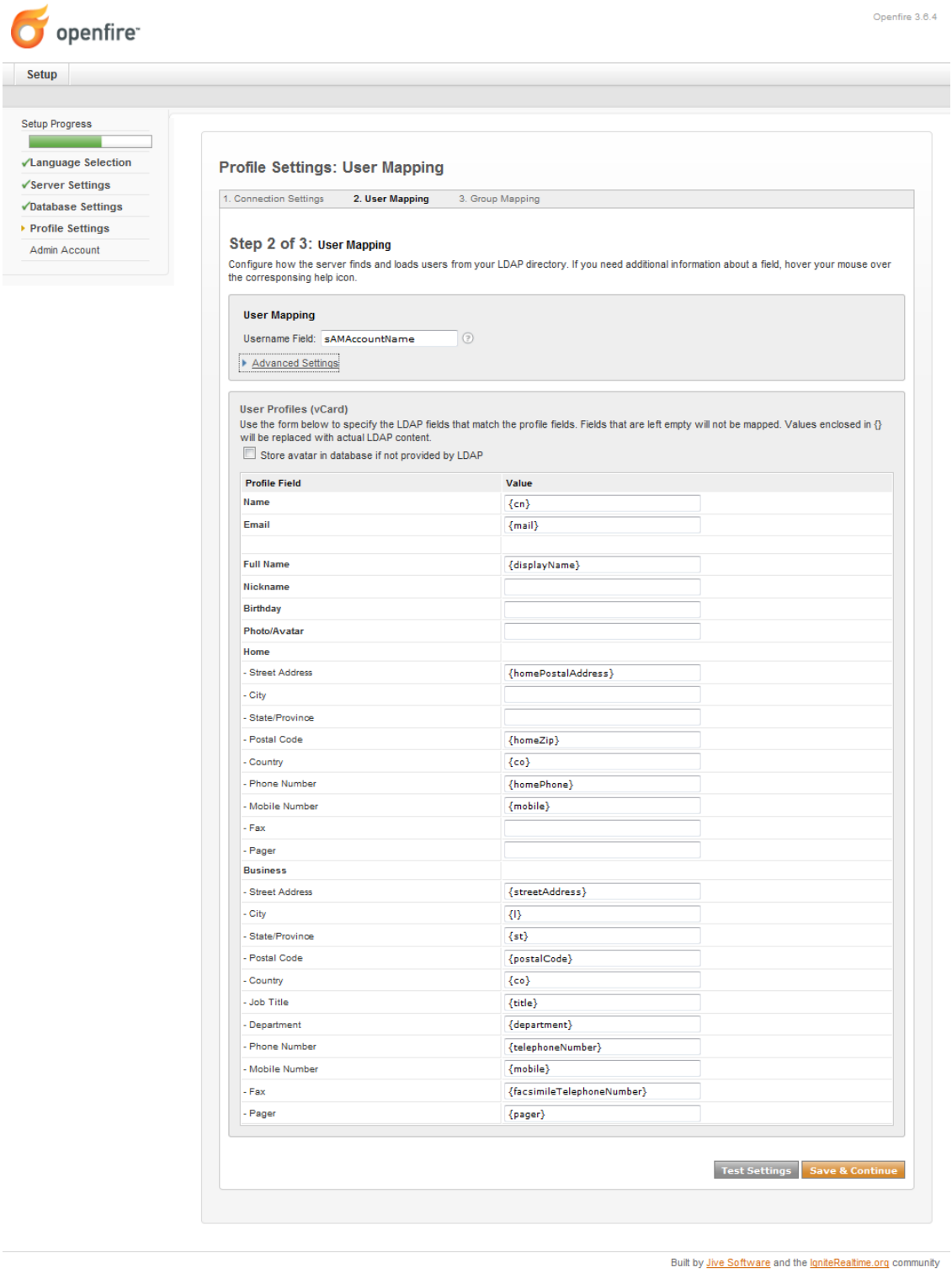
DTI. Fig 41.

6. Datos de conexión Directorio Activo:



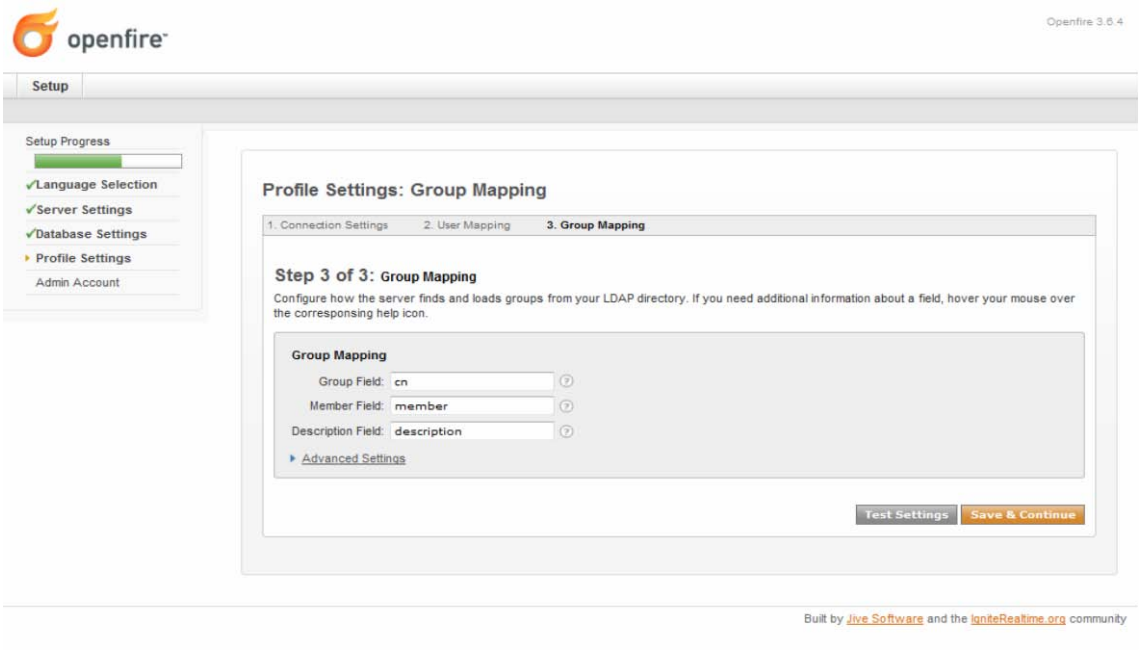
DTI. Fig 42.

7. Mapeo campos usuario Directorio Activo en Openfire:



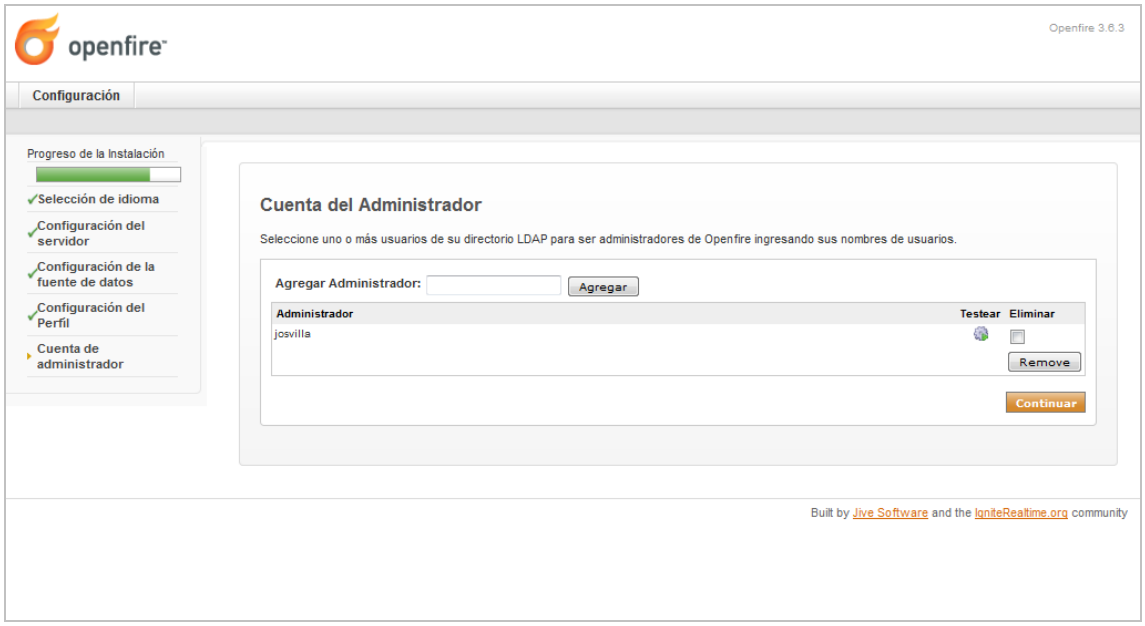
DTI. Fig 43.

8. Mapeo campos de grupo del Directorio Activo en Openfire:



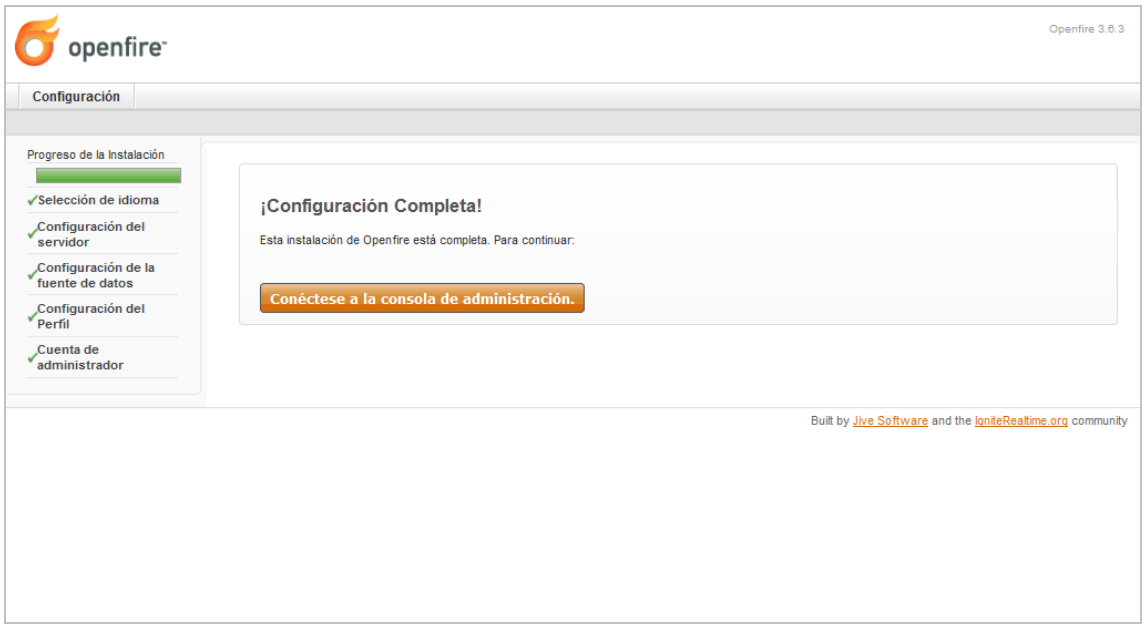
DTI. Fig 44.

9. Mapeo usuario administrador:



DTI. Fig 45.

10. Fin de la configuración:



DTI. Fig 46.

Importante dar de alta un usuario administrador y guardar sus datos, sino no podrá validarse ningún administrador en la consola de administración de openfire.

DT.I.10. Instalación Plugin SIP:

Este plugin se debe instalar tanto en el servidor, como en aquellos clientes en los que se quiera disponer de soporte SIP desde el cliente Spark.

La instalación en ambos extremos se describe a continuación:

Instalación plugin SIP en servidor Openfire:

- Descargamos el plugin de la página web de Igniterealtime:

  http://www.igniterealtime.org/projects/openfire/plugins/sip.jar

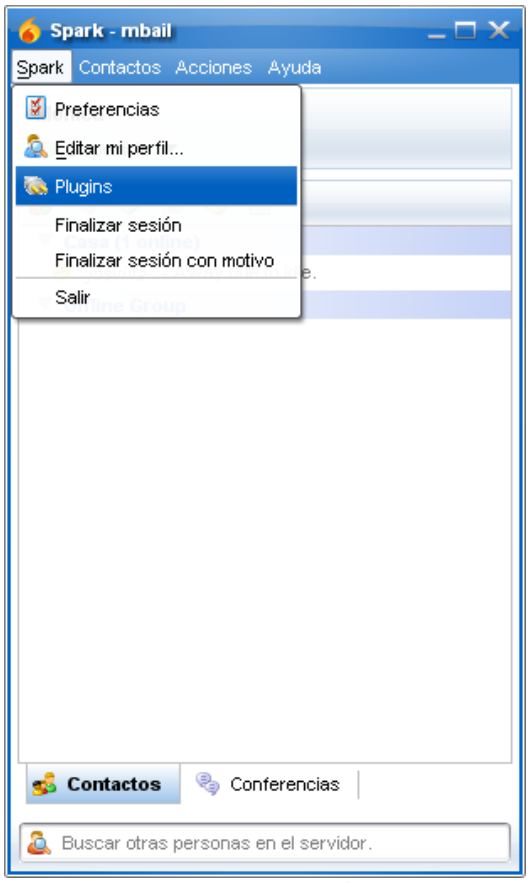
- Paramos Openfire.

- Se lanza un FTP desde nuestro Terminal hacía el path: **/opt/openfire/plugins** del servidor de Openfire y transmitiremos el fichero **sip.jar** que nos hemos descargado.

- Iniciaremos Openfire y tras el arranque habrá quedado instalado el plugin.

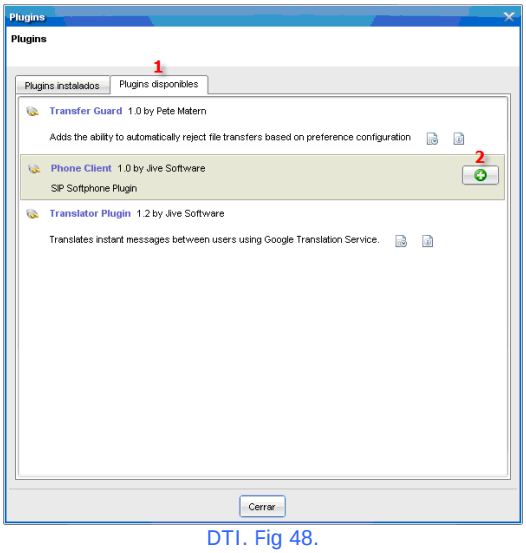
Instalación plugin SIP en cliente Spark:

1. Iniciamos la instalación del nuevo plugin:



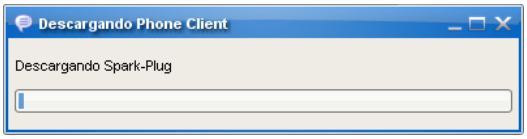
DTI. Fig 47. Pantallas instalación Plugin SIP.

2. Abrimos la pestaña de plugins disponibles y seleccionando el plugin "phone client", pulsamos el botón 



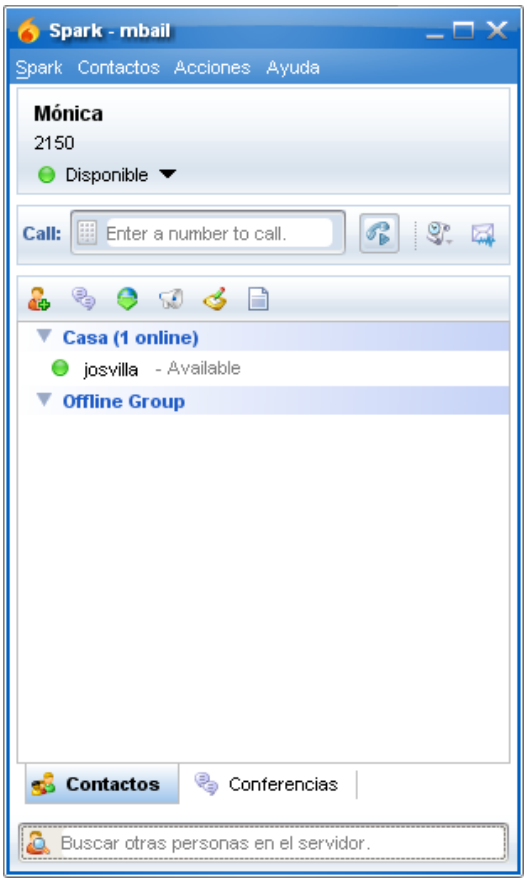
DTI. Fig 48.

3. El plugin se descargará e instalará de forma automática:



DTI. Fig 49.

4. Al finalizar la instalación, reiniciaremos sesión y ya estará instalado el plugin. Si el usuario jabber tiene mapeado correctamente un usuario SIP, al arrancar de nuevo el cliente Spark, aparecerá un softphone con su usuario SIP registrado:



DTI. Fig 50.

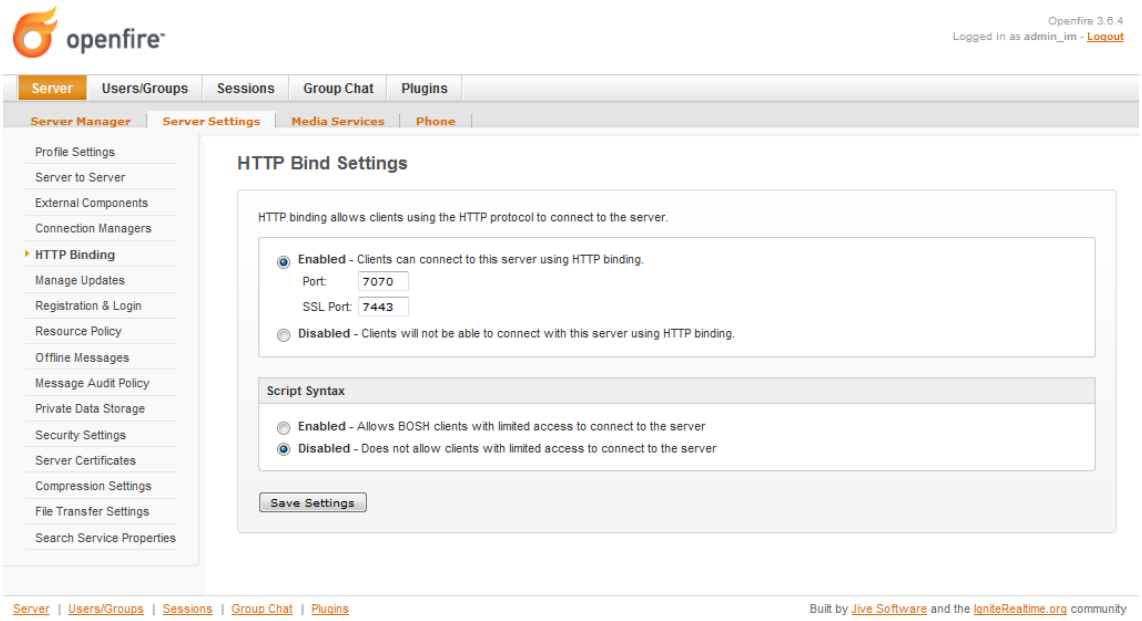
DT.I.11. Instalación Plugin Red5:

Antes de comenzar la instalación, debemos descargar el plugin desde la web de igniterealtime:

http://www.igniterealtime.org/projects/openfire/plugins /red5.war

Ahora, para su instalación seguiremos los siguientes pasos:

1. Habilitamos HTTP_BIND en openfire y anotamos el puerto que usa(el 7070 por defecto):



DTI. Fig 51. Http Bind Settings.

2. Paramos Openfire y copiamos el fichero **red5.war** en el path **/opt/openfire/plugins** Para copiar este fichero, podemos usar un software cliente FTP.

3. Arrancamos openfire. Tras arrancar openfire, el plugin se instalará automáticamente.

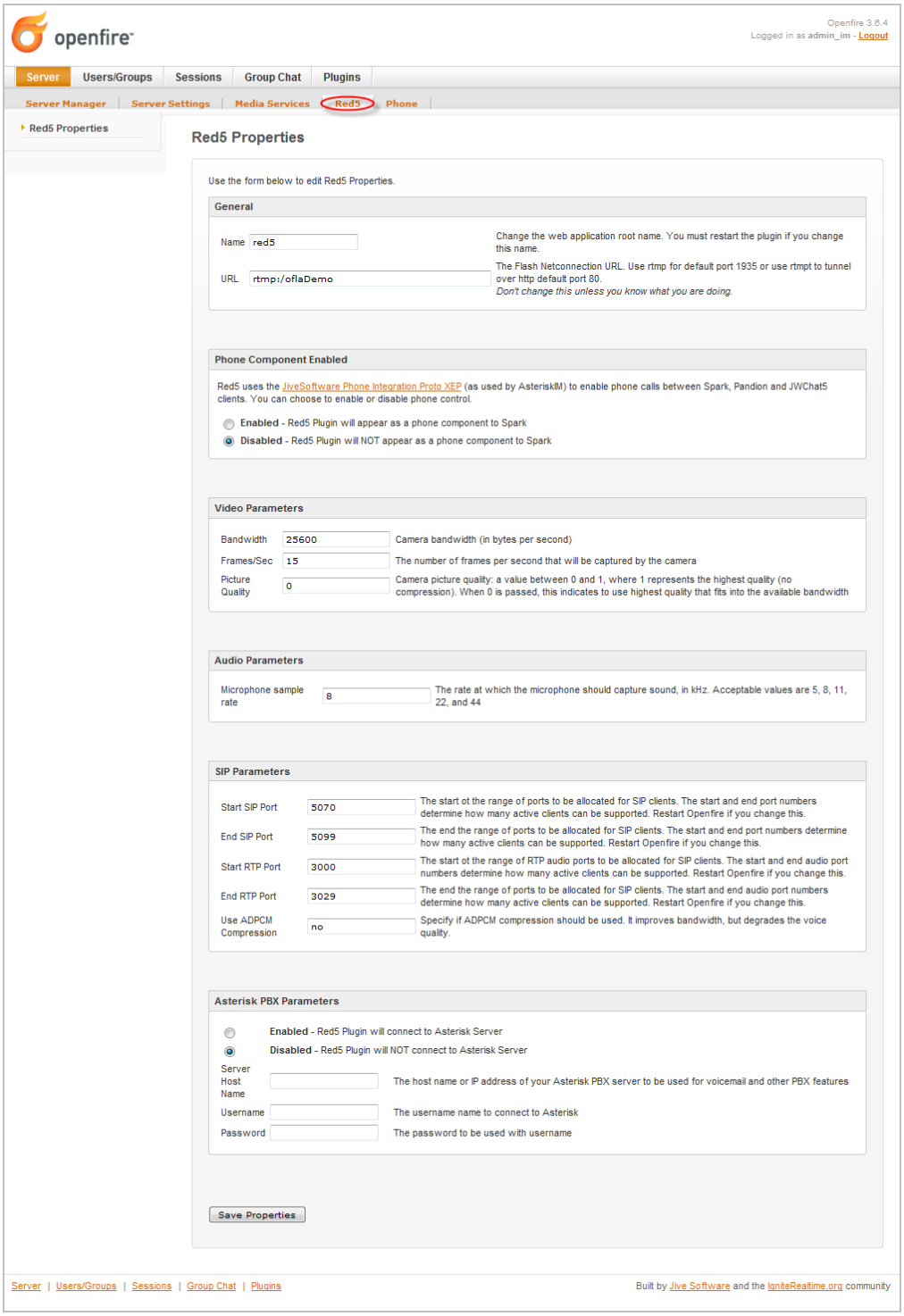
   Podremos validar que el plugin esta instalado cuando dentro del path **/opt/openfire/plugins** aparezca la carpeta **red5**:

```
root@imetcg:/opt/openfire/plugins# ls -l
total 32900
drwxr-xr-x  3 root     root          4096 2009-05-01 23:14 admin
drwxr-xr-x 15 root     root          4096 2009-11-29 12:37 red5
-rw-r--r--  1 joseluis joseluis 31866054 2009-11-29 12:35 red5.war
drwxr-xr-x  5 root     root          4096 2009-11-14 17:08 search
-rw-r--r--  1 root     root         29939 2009-05-01 23:12 search.jar
root@imetcg:/opt/openfire/plugins#
```

4. Se debe dar permisos de ejecución al fichero **asao2ulaw**. Este fichero se encuentra en este path: **/opt/openfire/plugins/red5/codecs**

5. Reiniciamos Openfire.

6. En la web de administración de Openfire, debe aparecer una pestaña más, desde donde se administrarán las propiedades de este plugin:



DTI. Fig 52. Administración plugin Red5.

7. Descargamos el plugin red5 para el cliente spark desde el propio servidor de Openfire, accediendo al enlace:

http://hostname:7070/red5/spark/red5-plugin.jar

8. Con el cliente Spark cerrado, copiamos el plugin en la carpeta donde se alojan los plugins: **C:\Archivos de programa\spark\plugins.**

9. Al iniciar sesión el plugin habrá quedado completamente instalado.

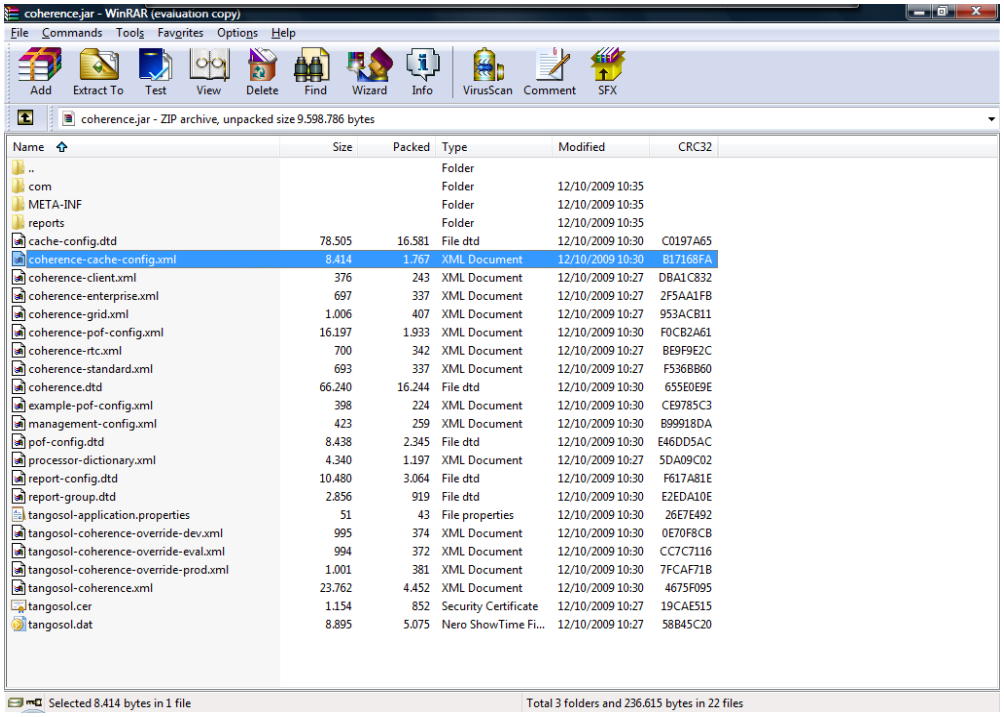
DT.I.12. Instalación Plugin Clustering:

1. Descargamos de la web de Oracle la versión 3.5.2 del Oracle Coherence for Java.

   Para descargar el fichero que contiene el Oracle Coherence for Java, se accederá a: http://www.oracle.com/technology/software/products/ias/htdocs/coherence.html
   Para poder iniciar la descarga debes de estar registrado previamente.

2. Una vez descargado el fichero, se abre el fichero zip que descargado, que en este caso ha sido este: “coherence-java-v3.5.2b463.zip”.

3. Acceder al path /coherence/lib, con un software de compresión de ficheros tipo winrar, se debe editar el fichero coherence.jar y borrar de dentro de su contenido el fichero coherence-cache-config.xml:



DTI. Fig 53. Edicion paquete coherente.jar.

4. Además del fichero modificado **coherence.jar**, en el path /coherence/lib encontraremos los ficheros **coherence-work.jar** y **tangosol.jar**. Copiaremos estos tres ficheros en el path /opt/openfire/lib del servidor de Openfire. Antes de copiar estos ficheros se deben de asignar permisos de escritura a la carpeta /lib de openfire.

   Para copiar estos ficheros podemos usar un software de transmisión de ficheros FTP tipo Filezilla.

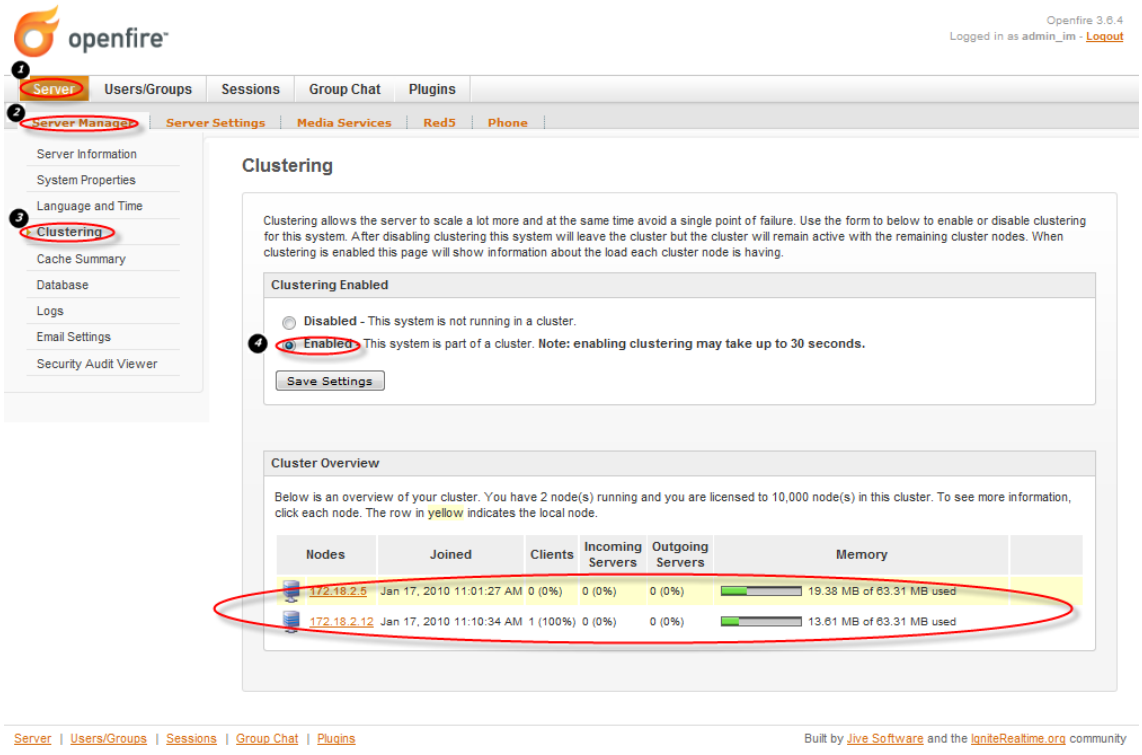
5. Reiniciar Openfire.

6. Descargamos de la web de IgniteRealtime el plugin clustering.jar, esta descarga se puede hacer desde este link: http://www.igniterealtime.org/projects/openfire/plugins/clustering.jar

7. Copiamos el fichero clustering.jar en el path /opt/openfire/plugins del servidor de Openfire. Previamente se deberán de asignar permisos de escritura a la carpeta /plugins. Para la copia de este fichero, también se puede usar un software tipo Filezilla.

8. Reiniciar Openfire.

Ahora solo queda habilitar el clustering en todos los nodos. Para habilitarlo, desde la consola de administración se accederá a Server, Server Manager, Clustering:



DTI. Fig 54. Clustering.

Cuando se habilita el plugin de clustering en una máquina, lo que hace este Openfire es enviar una petición multicast a todas las máquinas de su subred, si alguna de ellas es otro Openfire con el clustering habilitado, las dos se podrán a hablar e intercambiaran información de estado entre ellas.
Para aquellas instalaciones donde los servidores Openfire están en diferentes redes (como podría llegar a darse en el entorno de la UPC, donde pueden haber servidores Openfire en todos los campus), la petición multicast no nos serviría ya que en su subred no encontraría las máquinas de otros campus, en este caso podemos configurar el cluster para que enviara una petición unicast, especificando manualmente las direcciones IP o hostnames de cada nodo.

Para hacer esta configuración se debe editar el fichero "**tangosol-coherence-override.xml**" que se encontrará en el path: "**/opt/openfire/plugins/clustering**" de la instalación donde esta habilitado el clustering.

Este fichero contiene la siguiente configuración:

```
<coherence>

    <cluster-config>
        <multicast-listener>
          <port system-property="tangosol.coherence.clusterport">32386</port>
          <join-timeout-milliseconds>20000</join-timeout-milliseconds>
        </multicast-listener>

        <shutdown-listener>
          <enabled system-property="tangosol.coherence.shutdownhook">false</enabled>
        </shutdown-listener>

        <services>
            <service id="3">
                <init-params>
```

```
                    <init-param id="6">
                        <param-name>backup-count</param-name>
                        <param-value
system-property= "tangosol.coherence.distributed.backupcount">1</param-value>
                    </init-param>
                </init-params>
            </service>
        </services>

    </cluster-config>

    <configurable-cache-factory-config>
      <class-name>com.jivesoftware.util.cache.JiveConfigurableCacheFactory</class-name>
      <init-params>
        <init-param>
          <param-type>java.lang.String</param-type>
          <param-value       system-property="tangosol.coherence.cacheconfig">coherence-
cache-config.xml</param-value>
        </init-param>
      </init-params>
    </configurable-cache-factory-config>

</coherence>
```

La configuración multicast se debe sustituir por las siguientes líneas:

```
<unicast-listener>
   <well-known-addresses>
      <socket-address id="1">
         <address system-property="tangosol.coherence.wka1">host1</address>
         <port system-property="tangosol.coherence.wka1.port">8088</port>
      </socket-address>
      <socket-address id="2">
         <address system-property="tangosol.coherence.wka2">host2</address>
         <port system-property="tangosol.coherence.wka2.port">8088</port>
      </socket-address>
   </well-known-addresses>
</unicast-listener>
```

DT.I.13. Instalación Plugins Webchat y Fastpath:

Los plugins webchat y fastpath forman el paquete Fastpath y por lo tanto la instalación de estos plugins se ha de hacer de forma conjunta, el instalar uno de ellos sin instalar los dos a la vez no proporciona ningún servicio añadido.

Los pasos a seguir para su instalación son los siguientes:

1. Descargamos de la web de IgniteRealtime los plugins webchat.jar y fastpath.jar, estas descargas se pueden hacer desde estos enlaces:

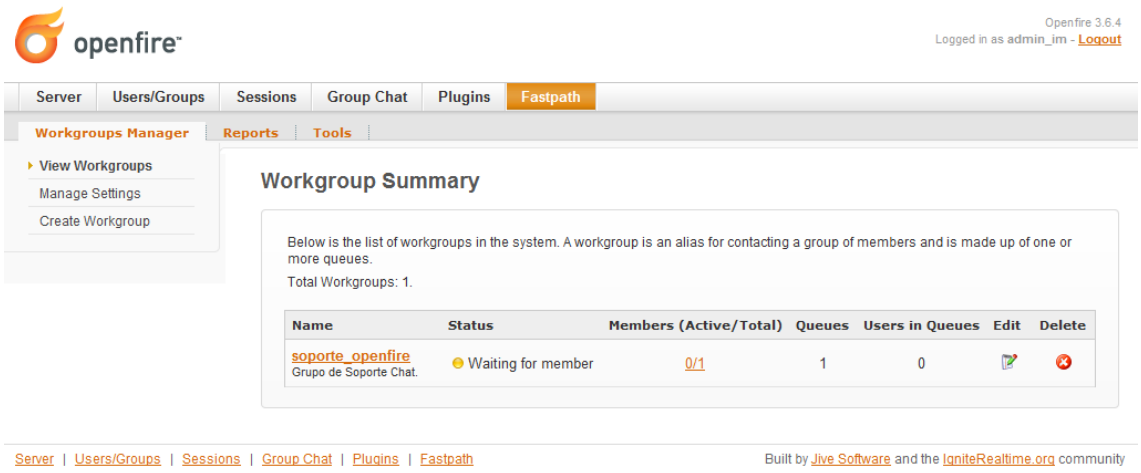
   http://www.igniterealtime.org/projects/openfire/plugins/webchat.war

   http://www.igniterealtime.org/projects/openfire/plugins/fastpath.jar

2. Con una herramienta cliente FTP, transferimos estos dos plugins a la carpeta /opt/openfire/plugins del servidor de Openfire.

3. Reiniciar Openfire y tras el arranque quedará instalado el paquete Fastpath.

En la consola de administración de Openfire, aparecerá una nueva pestaña bajo la cual se podrá hacer la configuración de grupos de trabajo y colas que se necesiten:



DTI. Fig 55. Administración plugin Fastpath.

DT.I.14. Instalación Plugin Kraken:

El plugin Kraken se encarga de habilitar la posibilidad de establecer pasarelas XMPP entre varios servidores externos, entre los cuales se encuentran Google Talk, MSN Messenger o MSN Yahoo.

El plugin está disponible en el repositorio del sitio web del projecto Kraken, también están disponibles todas las versiones del plugin, pero la última versión (1.1.2) la podemos descargar desde este enlace:

http://sourceforge.net/projects/kraken-gateway/files/kraken-gateway/1.1.2/kraken.jar/download

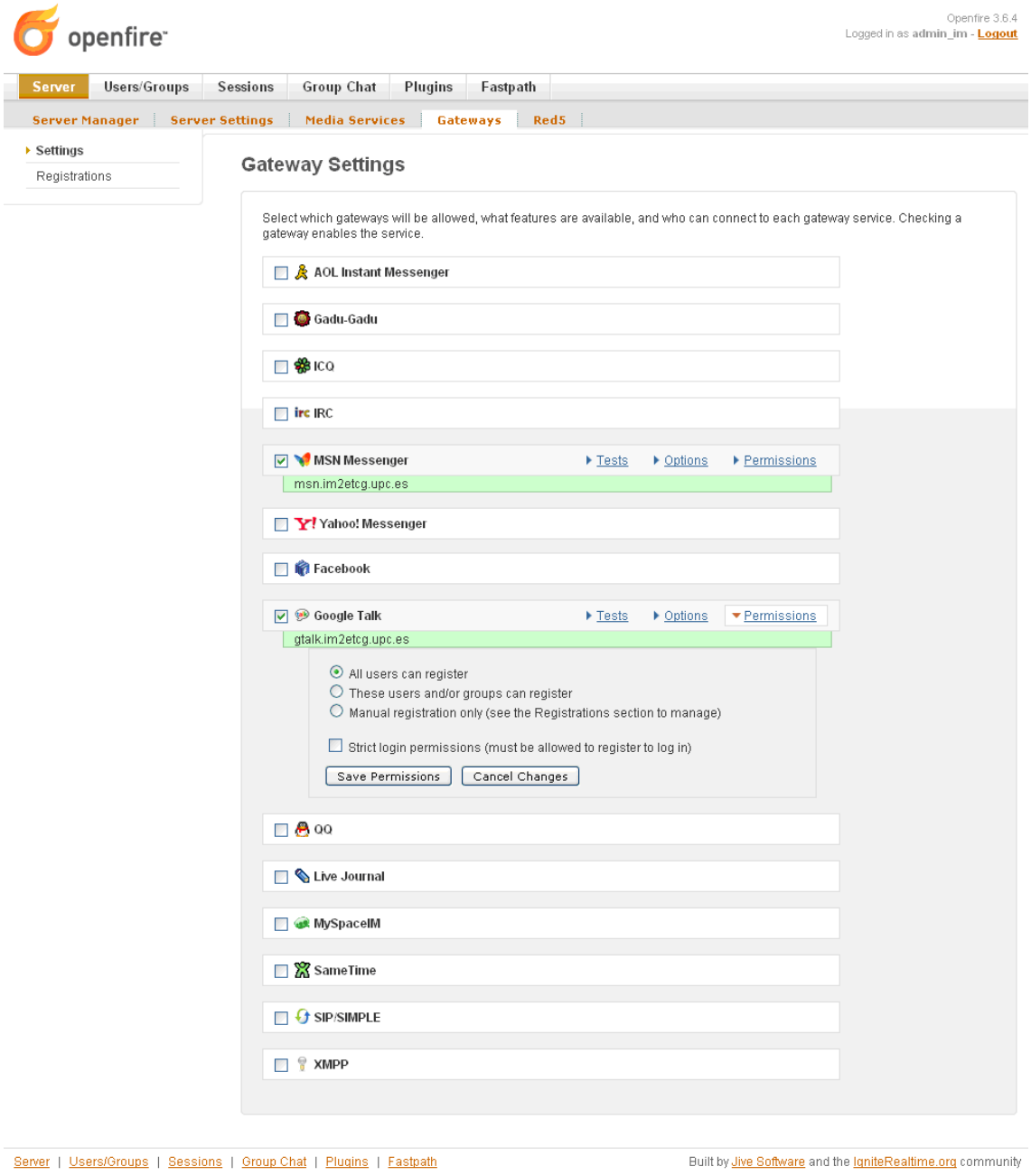
Los pasos a seguir para su instalación son los siguientes:

1. Descargamos la última versión del plugin Kraken.

2. Con una herramienta cliente FTP, transferimos este plugin a la carpeta, /opt/openfire/plugins del servidor de Openfire.

3. Reiniciar Openfire y tras el arranque quedará instalado el plugin.

En la consola de administración de Openfire, aparecerá una nueva pestaña bajo la cual se podrán habilitar las pasarelas soportadas por el plugin y hacer ciertos ajustes sobre los permisos de acceso a estas pasarelas:



DTI. Fig 56. Administración plugin Kraken.

DT.I.15. Compilación y creación instalable del cliente Spark personalizado:

Para la personalización del cliente Spark debemos editar y compilar el código fuente. Este código se puede descargar del siguiente servidor de Subversión:
http://svn.igniterealtime.org/svn/repos/spark/tags

Una vez editado y compilado necesitaremos crear un instalable con el nuevo código compilado.

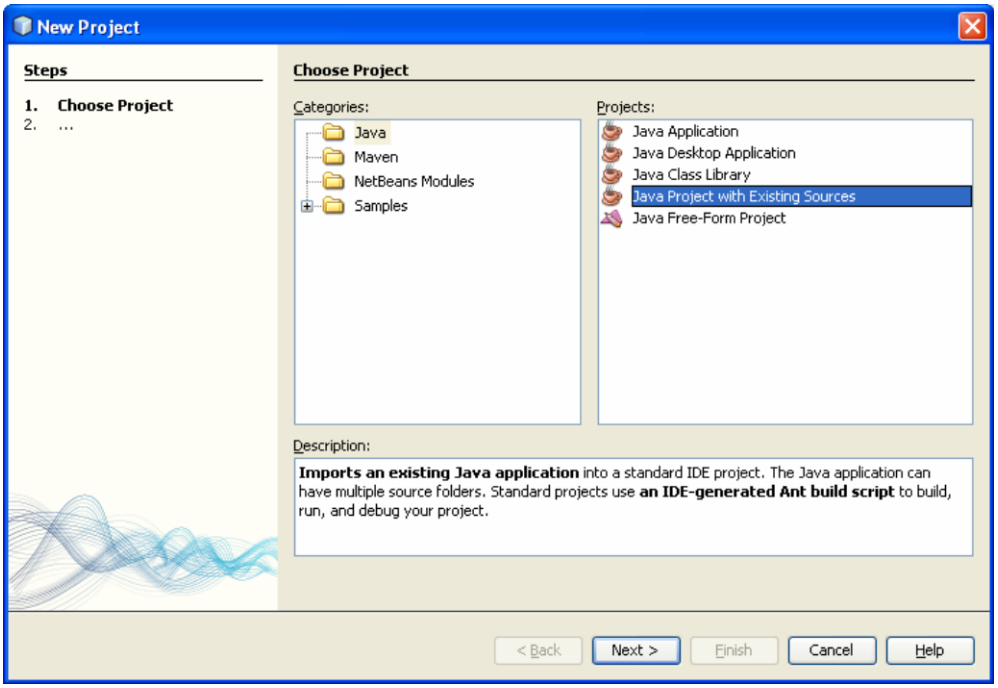
Estos dos procesos se resumen a continuación:

Edición y compilación del código fuente:

Para la edición y compilación se usará la aplicación NetBeans IDE 6.8. Esta aplicación requiere del "Java Developer Kit", ambos software se pueden instalar juntos a través del paquete "jdk-6u17-nb-6_8-windows-ml.exe" y se puede descargar directamente de la página web de sun microsystems,
http://java.sun.com/javase/downloads/widget/jdk_netbeans.jsp

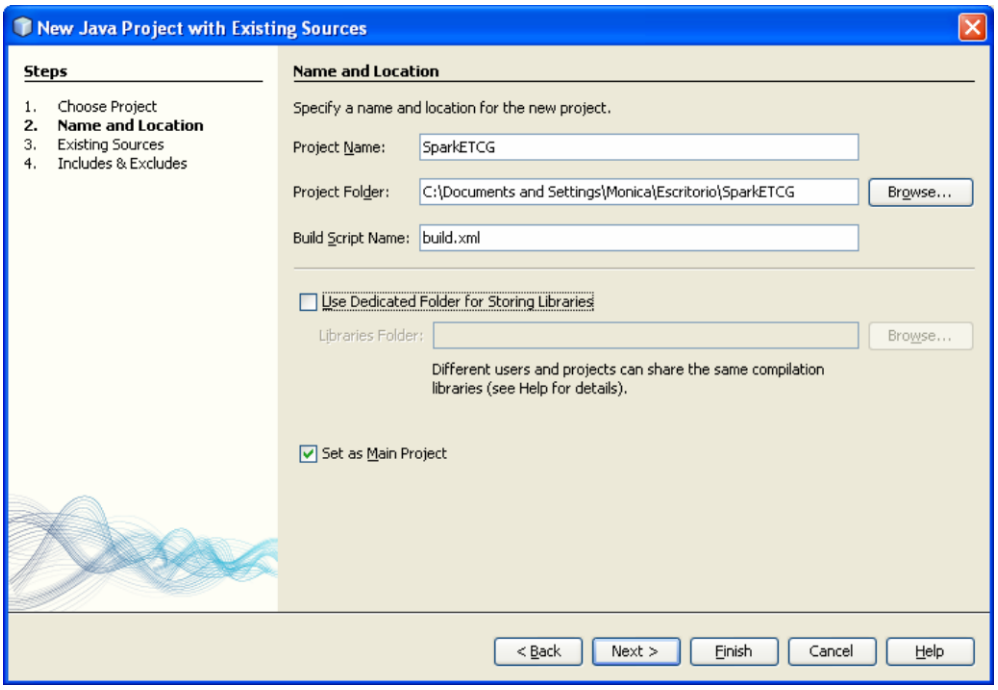
Los pasos son los siguientes:

1. Se crea un nuevo Proyecto tipo: "Java Project with Existing Source":



DTI. Fig 57. Pantallas compilación código fuente.

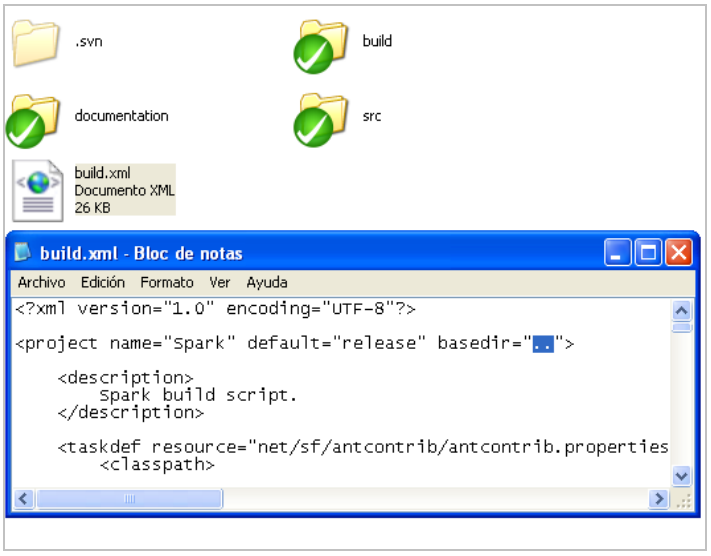
2. Se asigna nombre al proyecto y se indica el path donde se guardará este proyecto. En este path se copiarán los códigos fuente:



DTI. Fig 58.

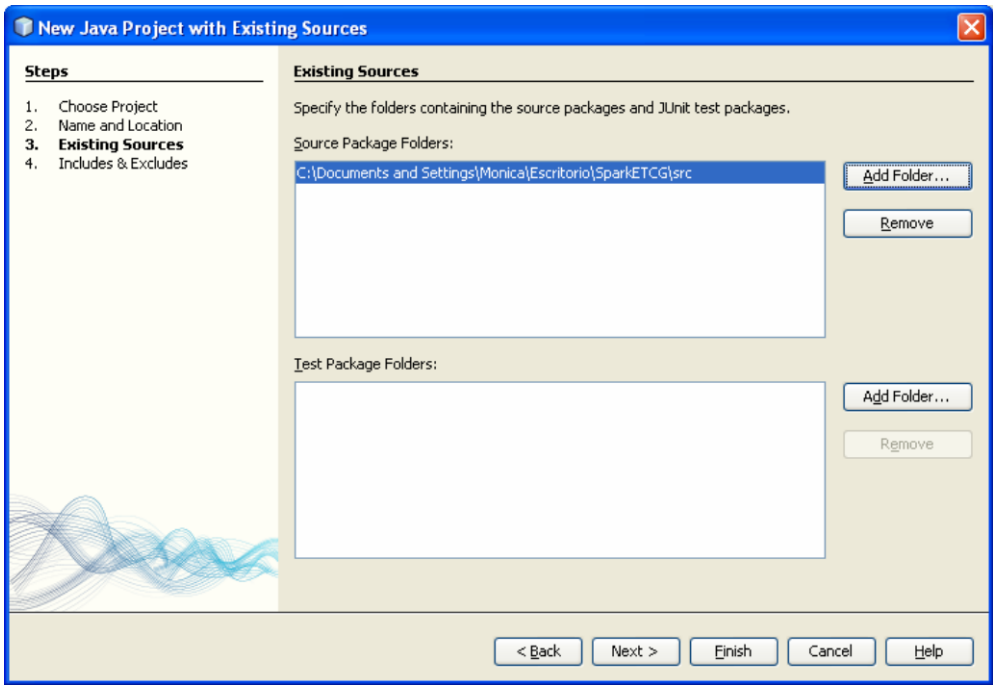
3. Se copia el fichero build.xml en la carpeta donde se aloja el proyecto. El fichero build.xml se puede encontrar en la carpeta: /build.

Se editará el fichero con el notepad y se eliminará el texto seleccionado:



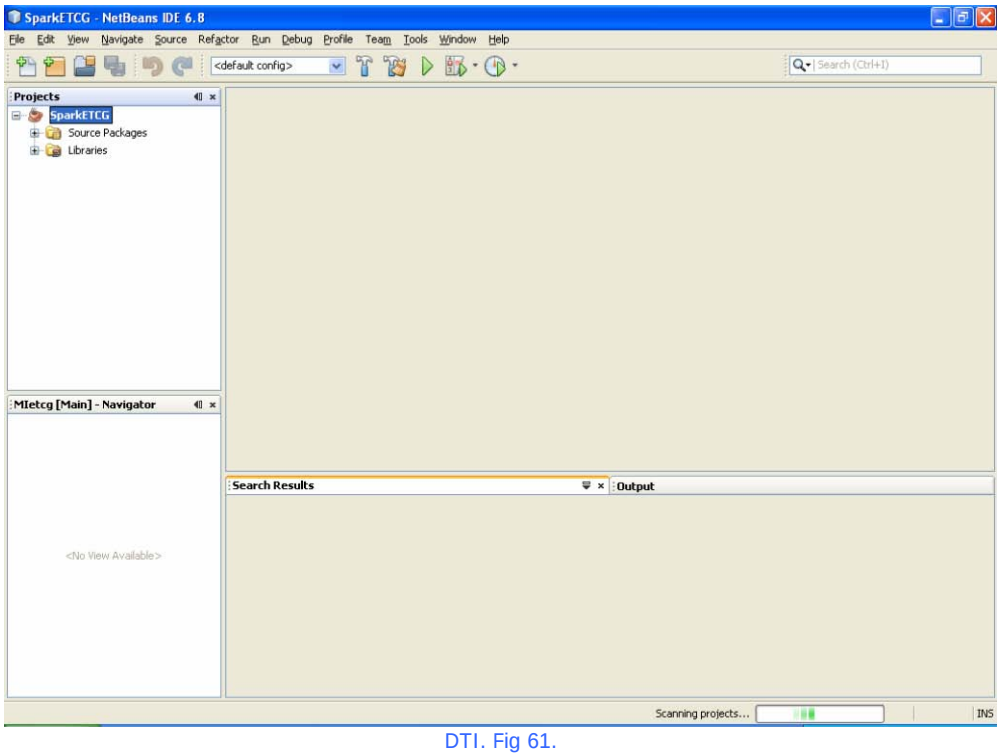
DTI. Fig 59.

4. Se añade la localización de donde se encuentra la carpeta /src con el código fuente y pulsamos “finísh":



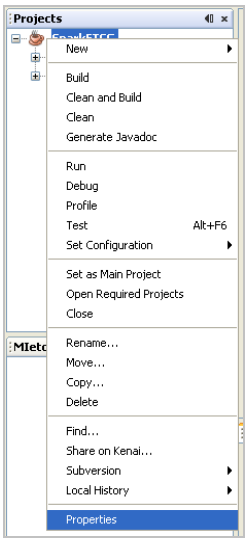
DTI. Fig 60.

5. En la parte izquierda aparecerá el icono de una taza de café con el nombre del proyecto:



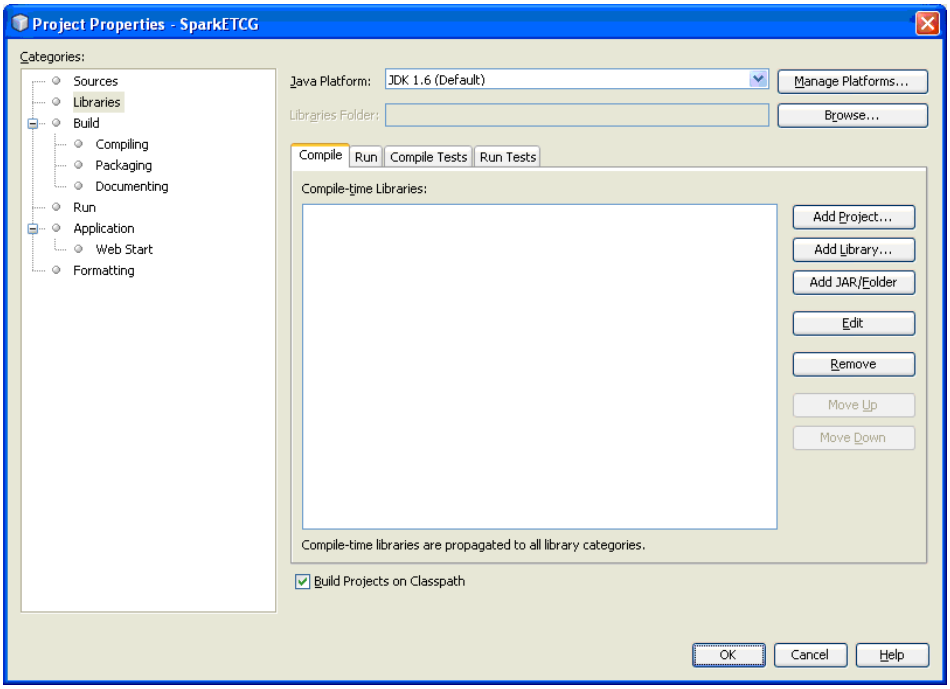
DTI. Fig 61.

Sobre este icono se hace clic con el botón derecho y se selecciona “Properties”:



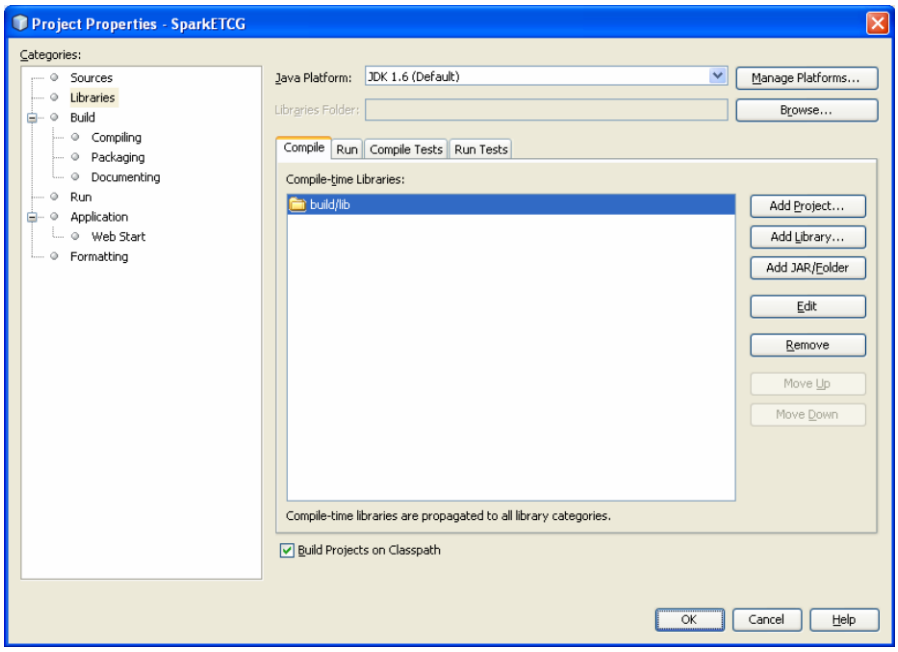
DTI. Fig 62.

6. Se abrirá la siguiente pantalla en la que se selecciona la opción de “Libraries”:



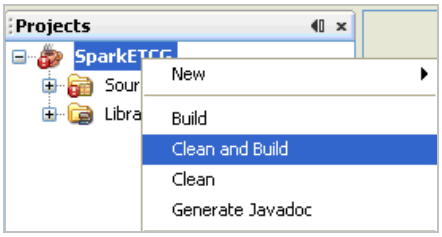
DTI. Fig 63.

Haremos clic en “Add JAR/Fólder” y se indicará la localización del path “/build/lib”:



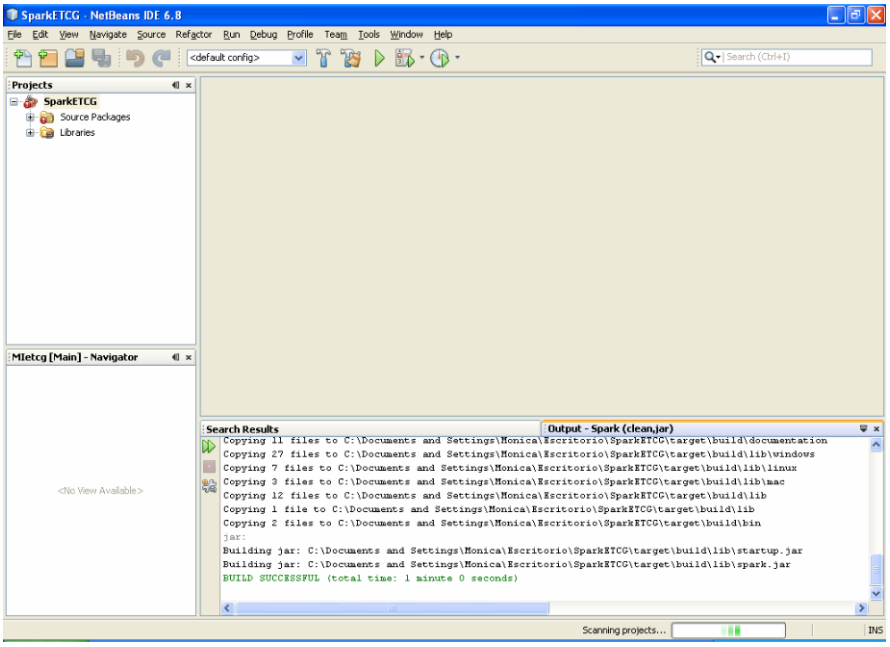
DTI. Fig 64.

7. Sobre el icono de la taza de café, se hace clic con el botón derecho y se selecciona “Clean and Build”, se iniciará la compilación:



DTI. Fig 65.

8. El resultado de la compilación será el siguiente:



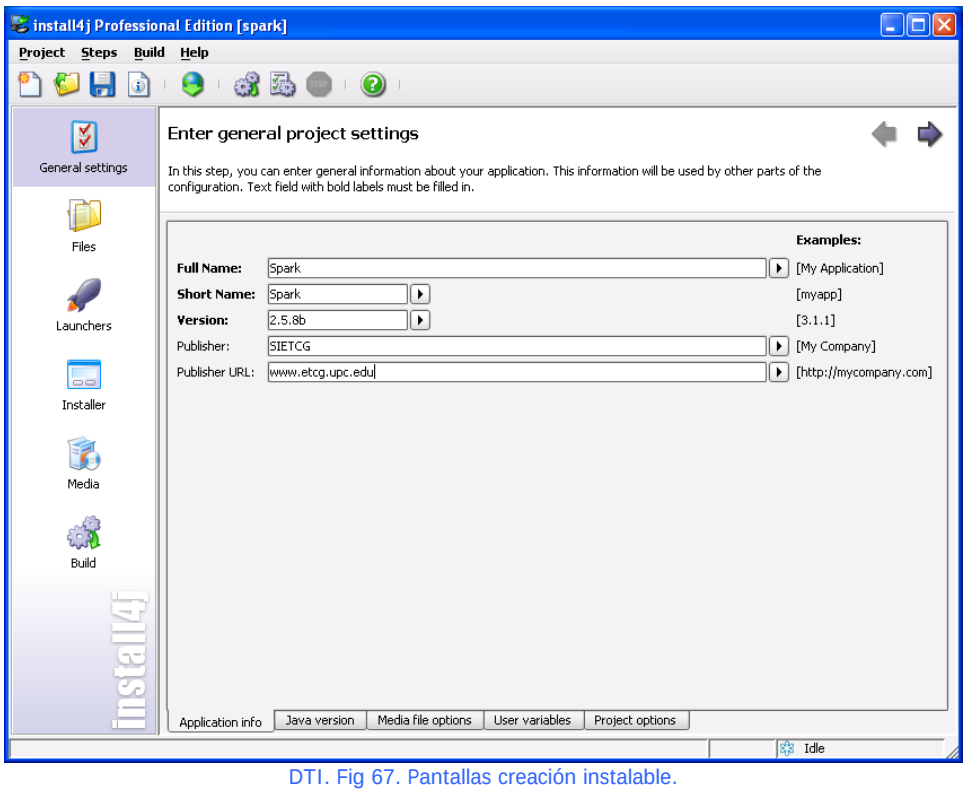
DTI. Fig 66.

Creación instalable del cliente Spark:

Para tener un instalable con el código modificado es necesario tener un software capaz de generar este instalable. El software seleccionado ha sido “install4j” de EJ-Technologies, se trata de un wizard que partiendo del código fuente compilado y personalizando una serie de opciones acaba generando un paquete de instalación.
A diferencia del software anterior, este último no es gratuito, pero desde la página web del fabricante se ha podido descargar una versión de evaluación de 90 días. Esta versión de evaluación se puede descargar desde este link:
http://www.ej-technologies.com/download/install4j/files.html

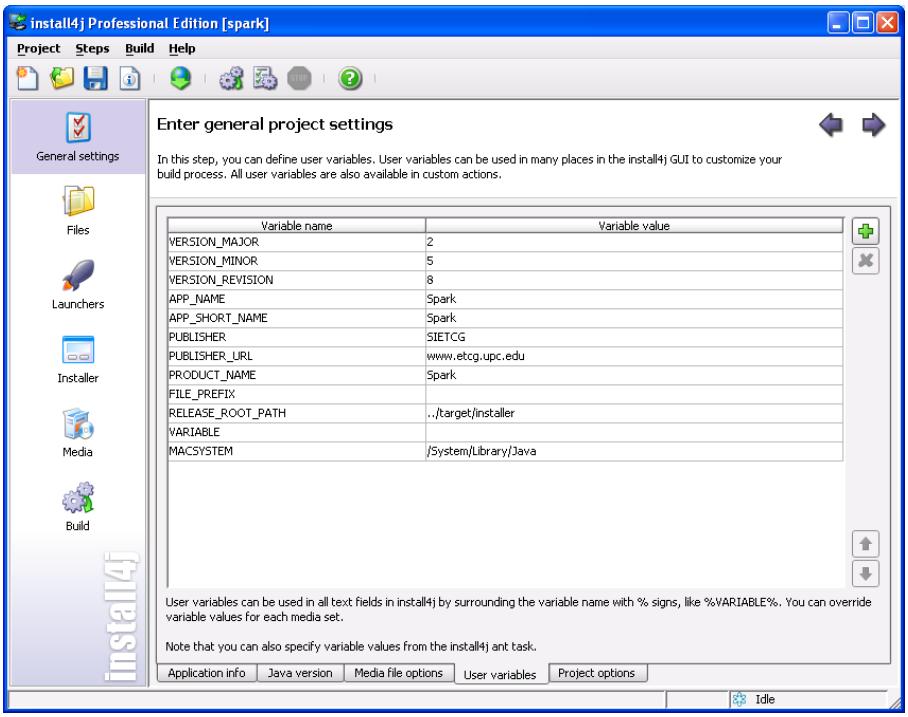
1. Con esta aplicación abriremos el proyecto que se localiza en el path: “builder\installer”.

2. Dentro de “General Settings”, iremos a la pestaña de “Application Info” y se podrán personalizar los siguientes parámetros:



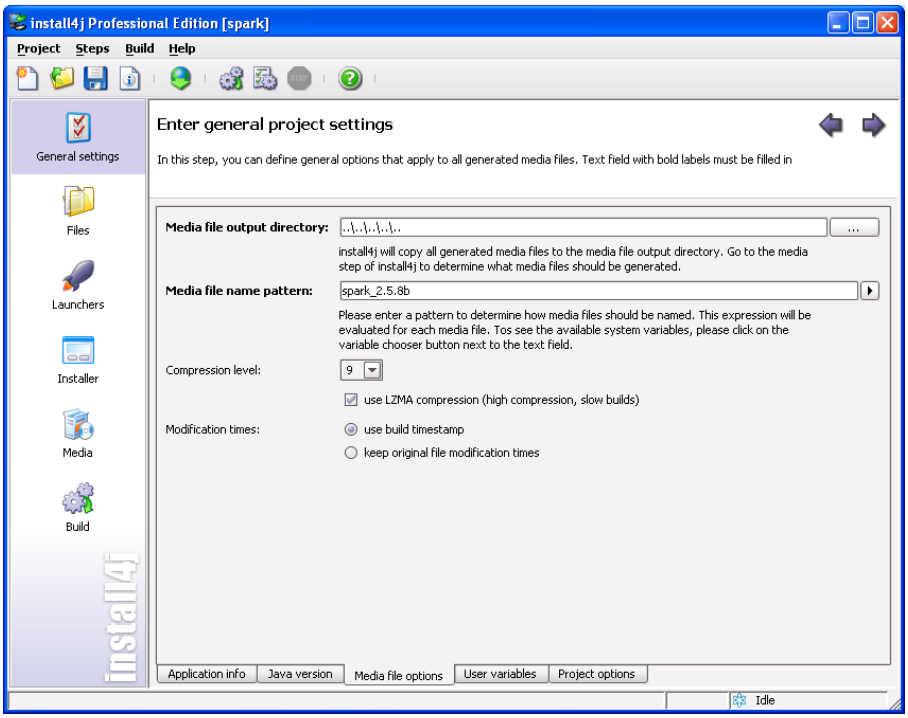
DTI. Fig 67. Pantallas creación instalable.

3. Dentro de “General Settings”, iremos a la pestaña de “User Variables” y se podrán personalizar los siguientes parámetros:



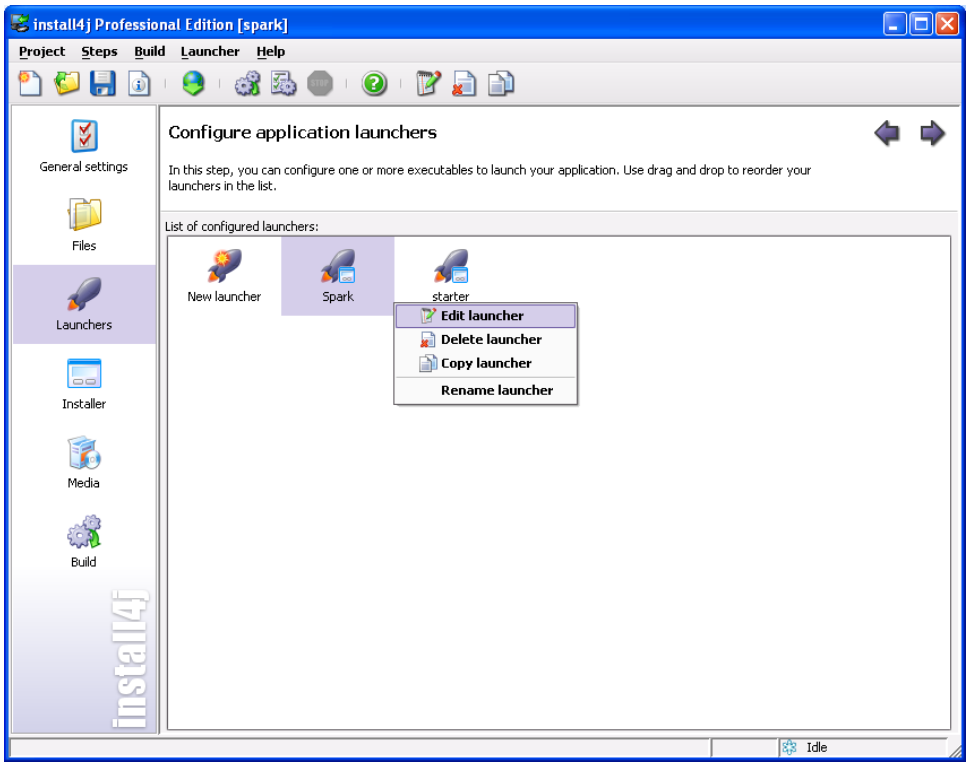
DTI. Fig 68.

4. Dentro de “General Settings”, iremos a la pestaña de “Media File Options” y se podrán personalizar los siguientes parámetros:



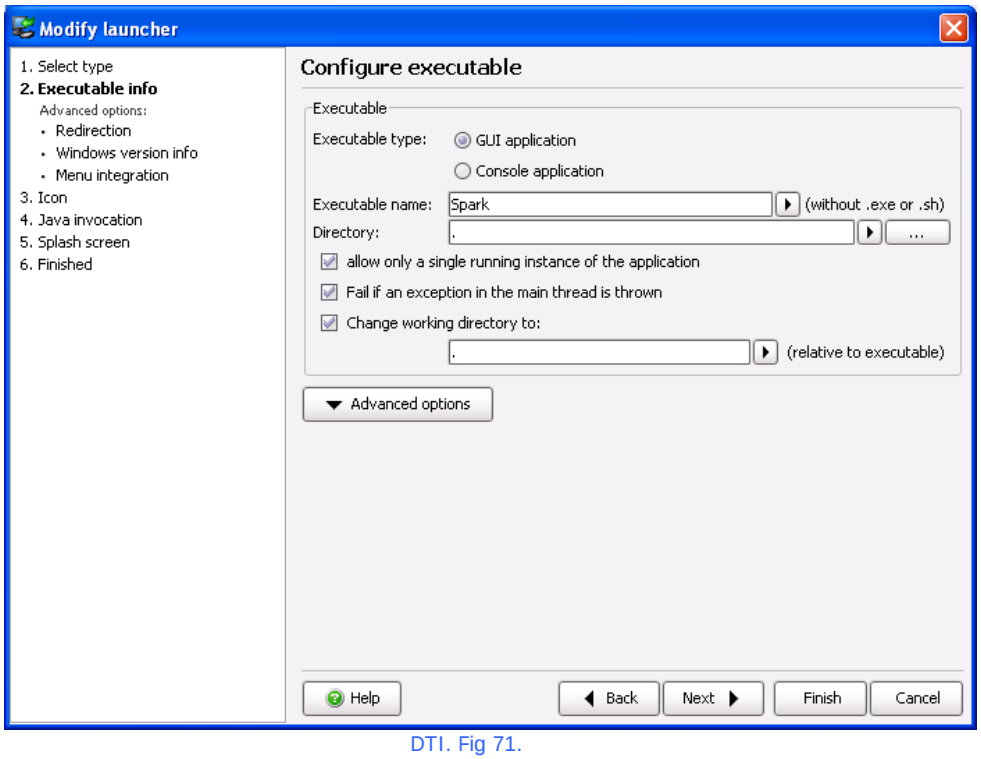
DTI. Fig 69.

5. En el menú de Launcher, podremos editar las propiedades del instalable:



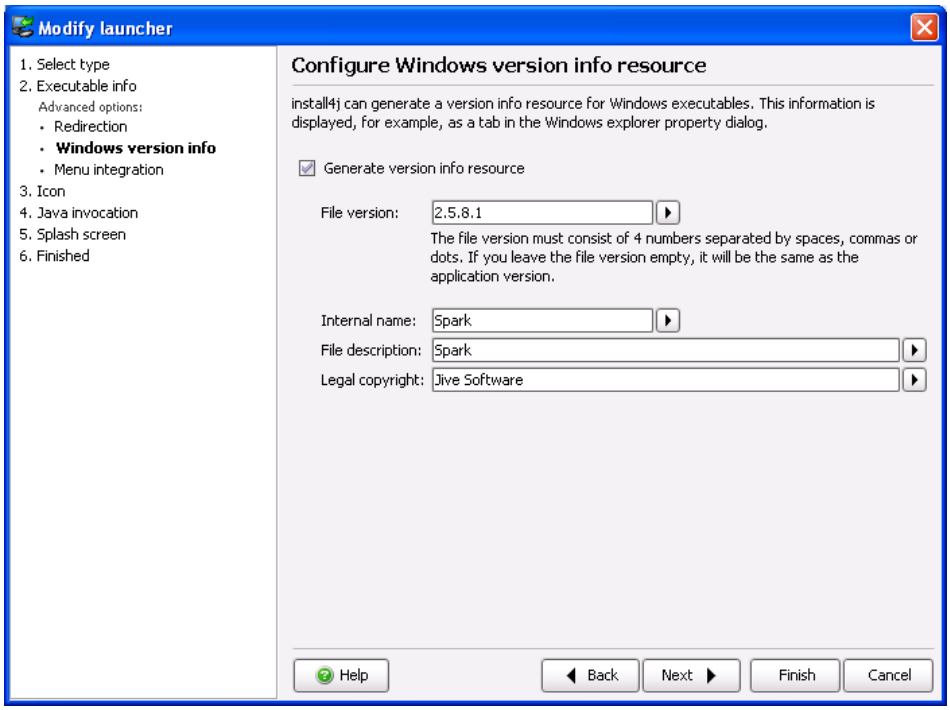
DTI. Fig 70.

6. En el menú “Executable Info” editaremos el nombre del instalable si se deseara cambiar:



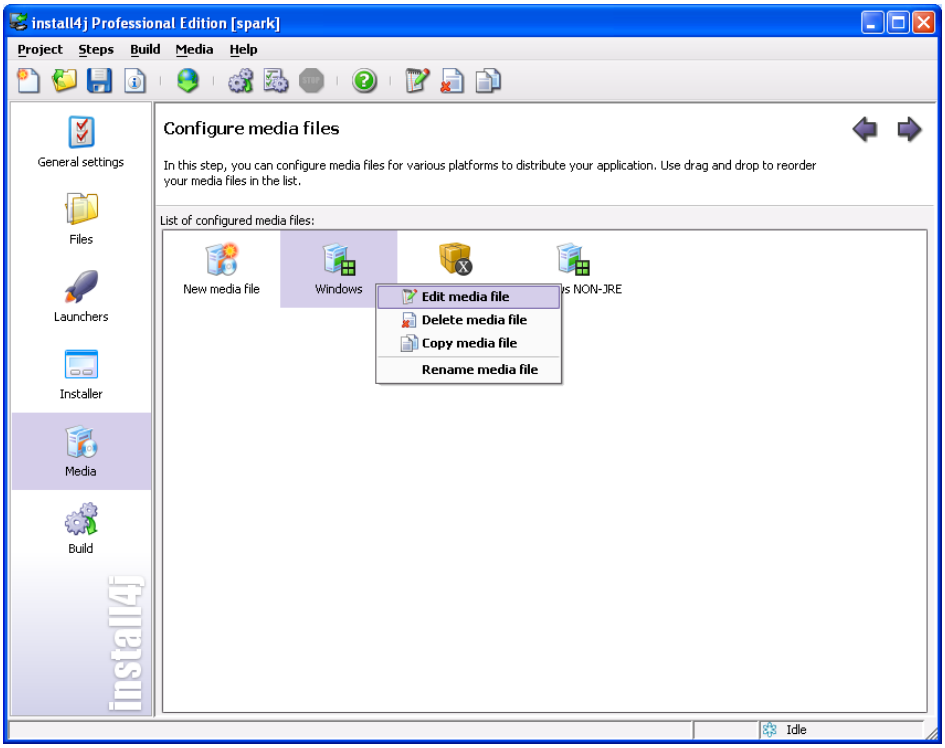
DTI. Fig 71.

7. En el menú “Executable Info, Windows version info” podremos editar la versión del nuevo instalable:



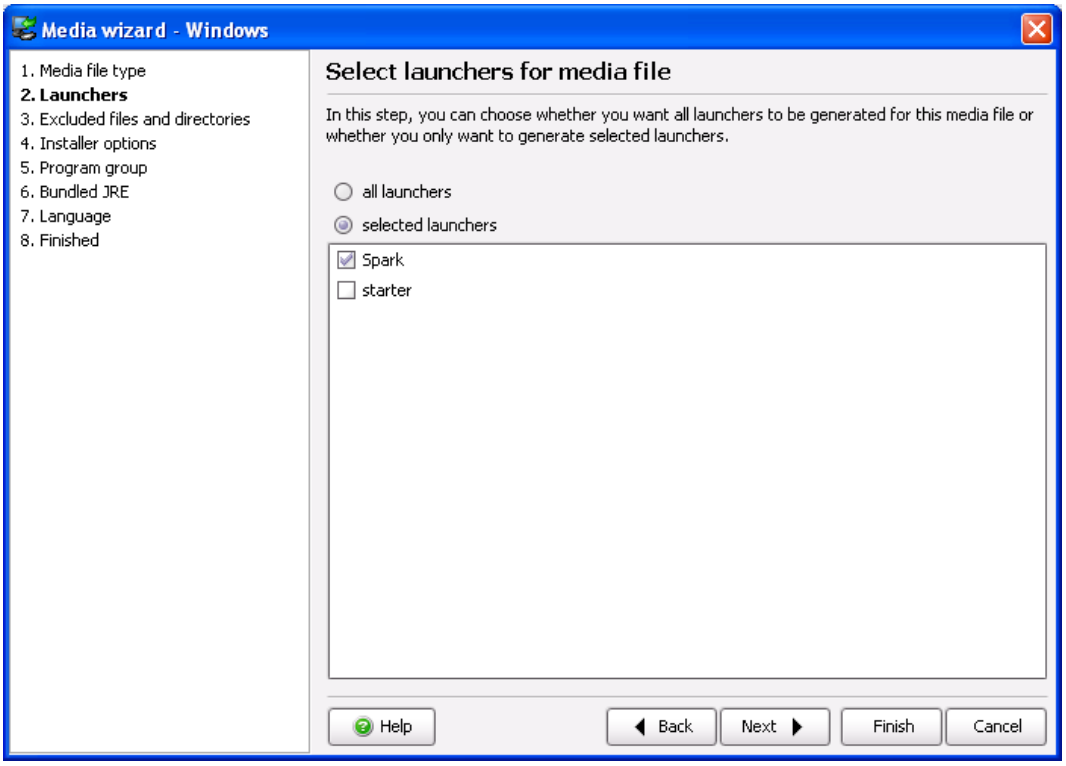
DTI. Fig 72.

8. En Media, en este caso editaremos las propiedades de Windows ya que al tratarse de una versión de evaluación solo permite generar wizards de instalación para Windows, pero en una versión licenciada se podrían editar los Media de Linux y MAC.



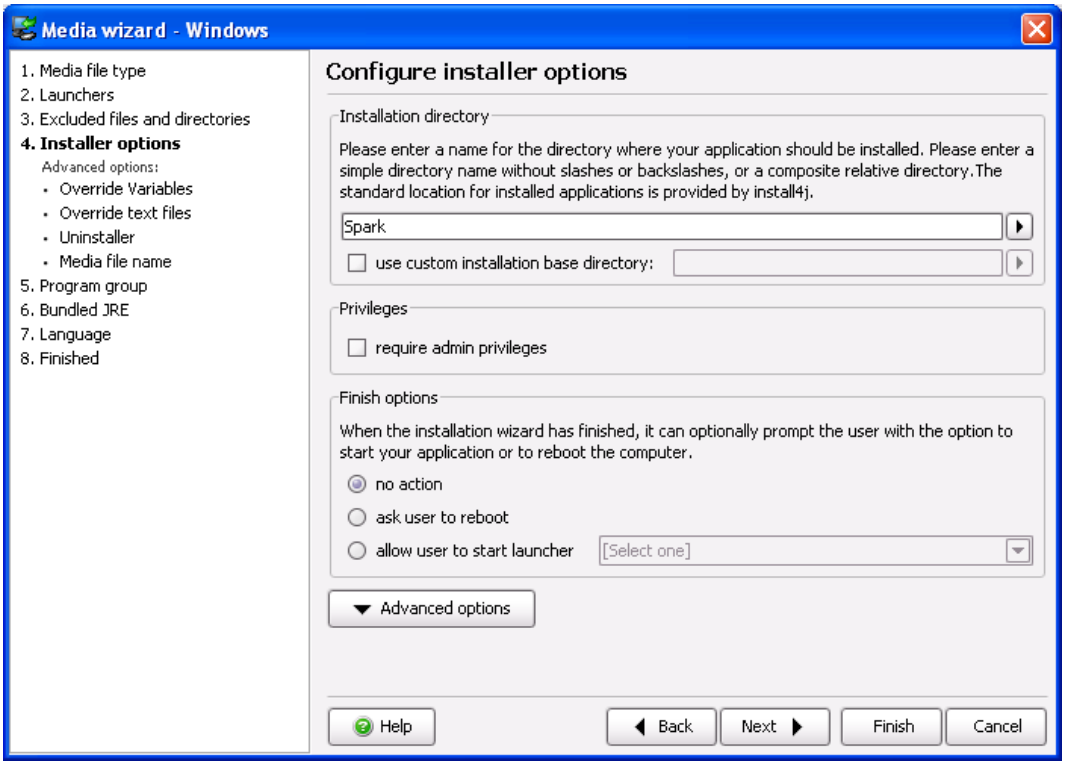
DTI. Fig 73.

9. Se selecciona el Launcher Spark y se pulsa Next:



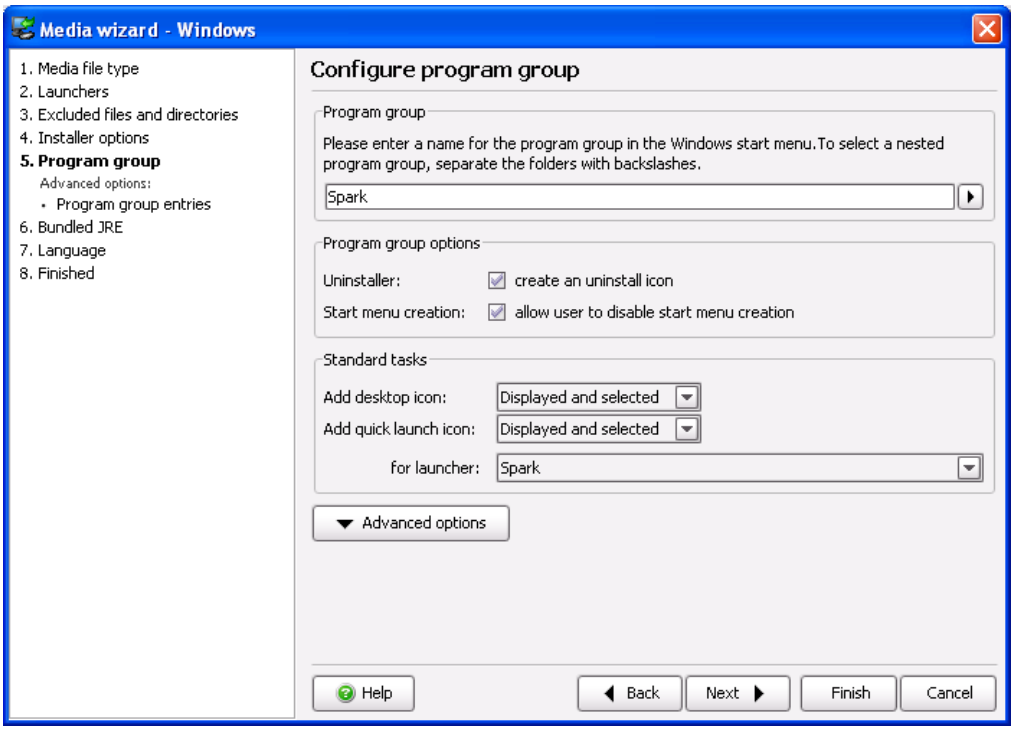
DTI. Fig 74.

10. Bajo la opción de “Installer Options” se puede editar el nombre del directorio de instalación de Spark, después se debe pulsar Next:



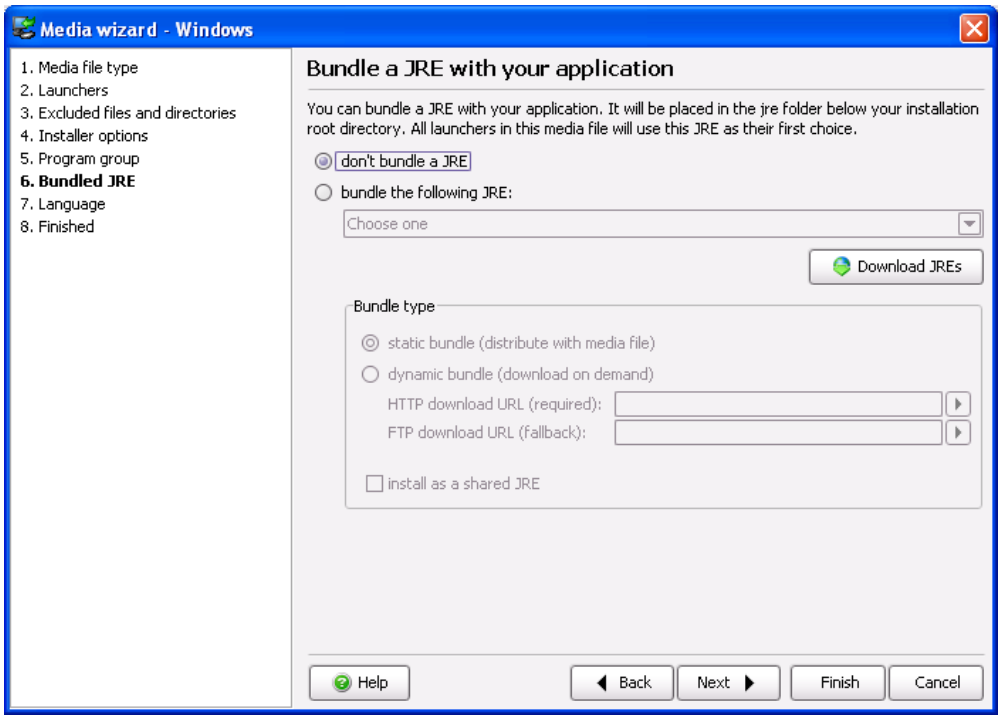
DTI. Fig 75.

11. Bajo la opción de “Program Group” se pueden editar las propiedades de instalación, tales como la creación de iconos o creación del paquete de desinstalación. Tras las elecciones se debe pulsar Next:



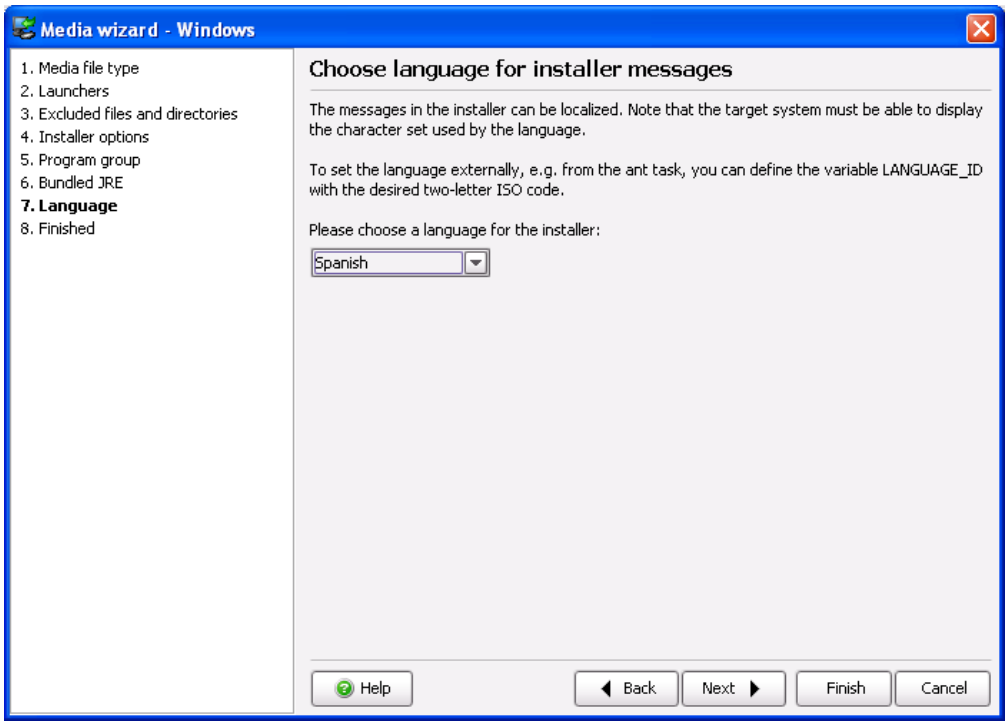
DTI. Fig 76.

12. Bajo la opción de “Bundled JRE”, podemos seleccionar una versión de la máquina virtual Java a instalar con el mismo paquete, ya que para que funcione Spark se requiere de la versión 1.6 o superior. Tras elegir si se instala la máquina virtual se debe pulsar Next:



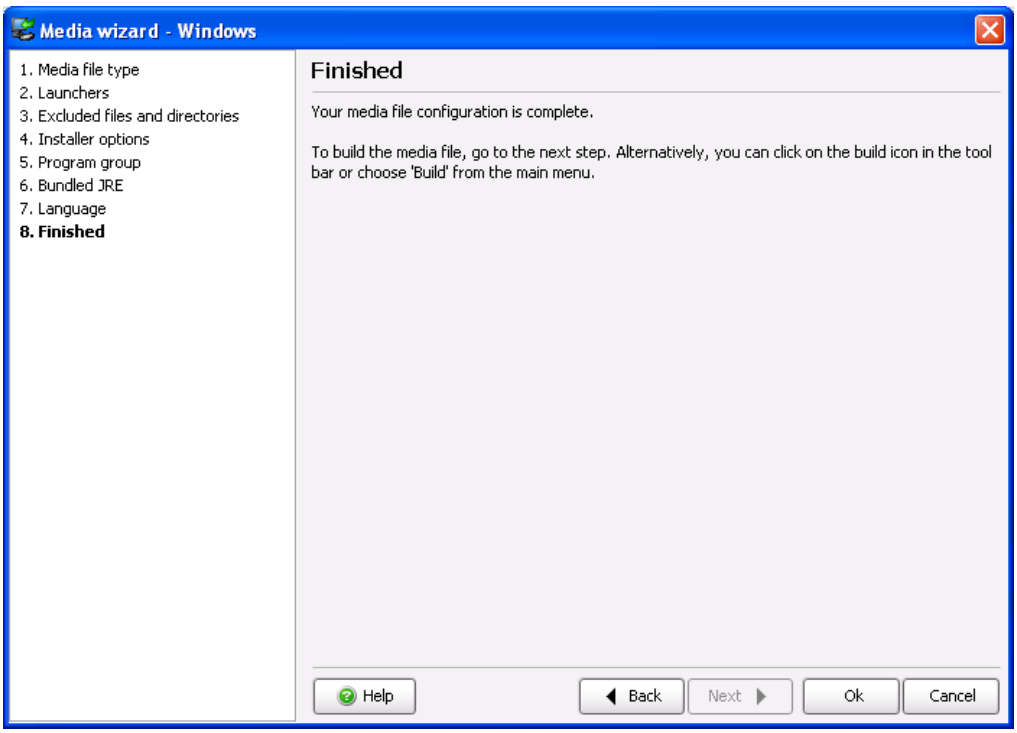
DTI. Fig 77.

13. Bajo la opción de Language, se puede seleccionar el idioma del paquete de instalación, para el caso de ETCG se ha seleccionado el Español, el Catalán no estaba disponible. Tras elegir el idioma se debe pulsar Next:



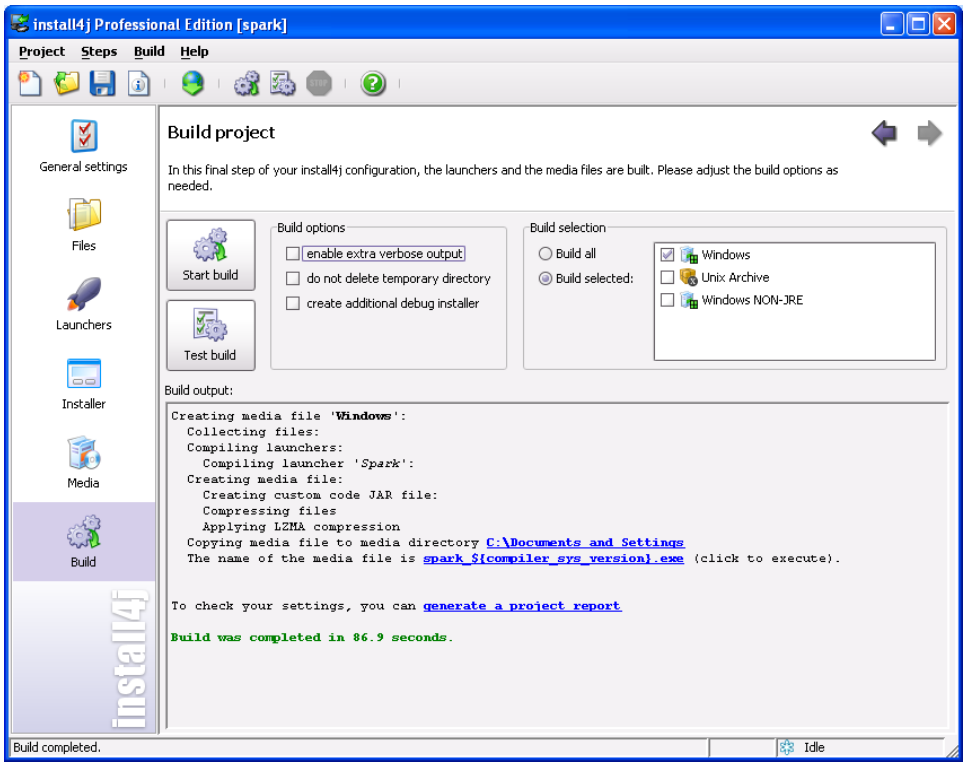
DTI. Fig 78.

14. Al llegar a esta opción pulsaremos OK:



DTI. Fig 79.

15. Bajo la opción de Build iniciaremos la creación del instalable pulsando el botón de Start Build:



DTI. Fig 80.

Documentos técnicos de usuario:

DT.U.01. Manual de usuario cliente Spark:

¡Bienvenido a Spark!, esta guía le proporcionará los conocimientos básicos para usar Spark. Podrá conectarse al sistema de mensajería, iniciar sesiones chat de texto, voz y vídeo e intercambiar ficheros.

Conectarse con Spark:

La conexión con Spark es muy sencilla, los pasos son los siguientes:

1. Arrancar Spark.
2. Introducir la información de Login y password, serán los mismos credenciales que los usados para registrar el PC en el dominio. Una vez introducidos estos credenciales, pulsar el botón de Login:

DTU. Fig 1. Cliente Spark ETCG.

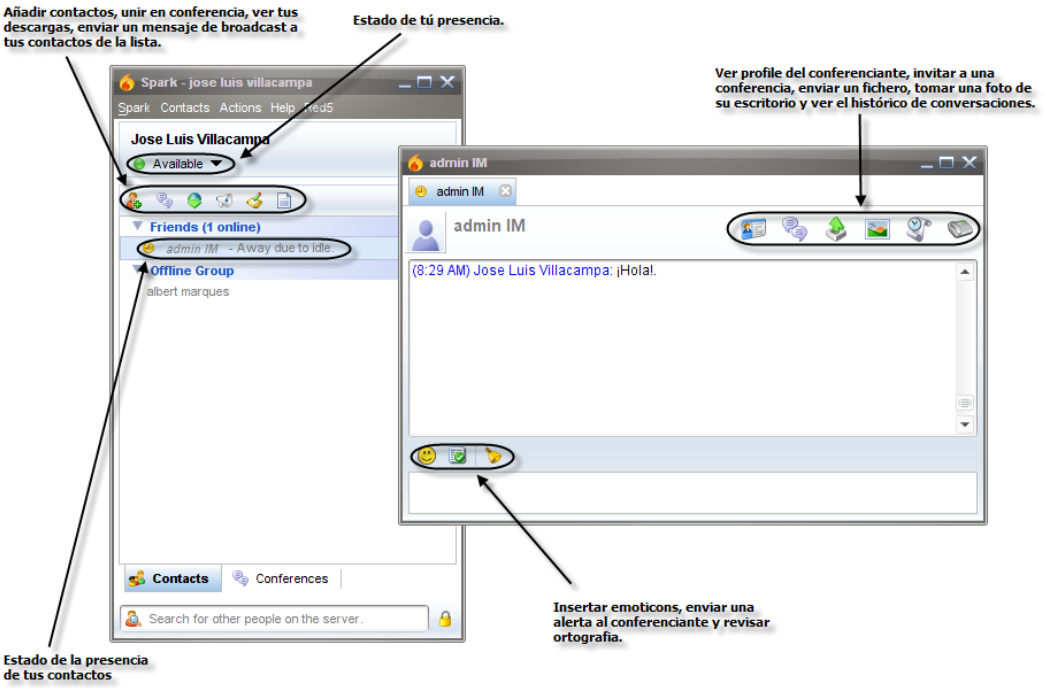
La gestión de usuarios se hace de forma centralizada desde el directorio activo del departamento, por lo tanto la opción de “Accounts” no permite crear nuevas cuentas de usuario.

Bajo la opción de “Advanced” quedarán los parámetros por defecto.

Conociendo Spark:

Si has usado aplicaciones de mensajería instantánea antes, encontrarás todo muy familiar y sino las has usado, encontraras que es muy fácil usar Spark.

Una vez conectado con Spark, pegue un vistazo a la pantalla principal. Después de que aparezca la lista de contactos podrás comenzar a iniciar sesiones de chat, encontrará las cosas que necesita de forma fácil y rápida.



DTU. Fig 2. Aspecto funcional Spark.

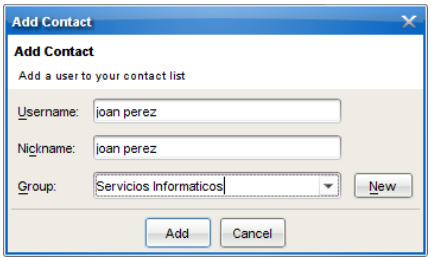
Añadir contactos:

Deberás añadir los contactos con los que necesitas contactar habitualmente. Para ello se deben seguir los siguientes pasos:

1. Haz clic en el botón de Add Contact:

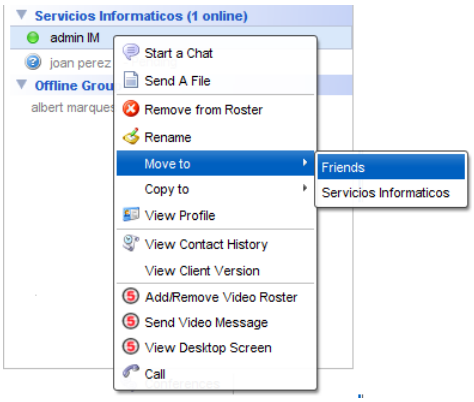
DTU. Fig 3.

2. Se abrirá la siguiente pantalla que rellenaremos con los datos de la persona que queremos añadir y pulsaremos el botón “Add”.



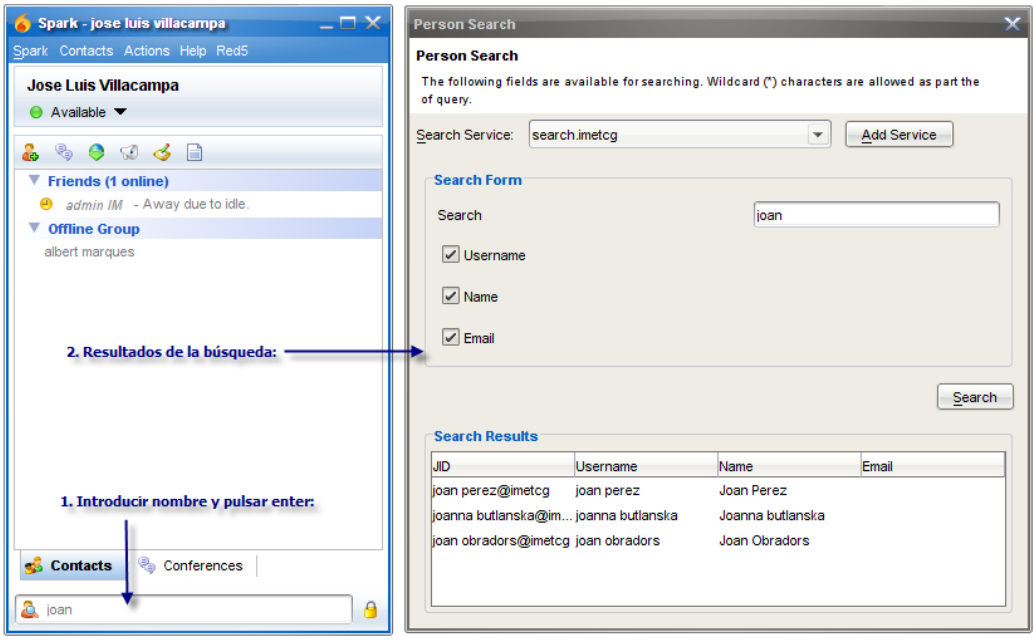
DTU. Fig 4.

Si usted decide que prefiere tener el contacto en otro grupo, simplemente haga clic sobre el contacto con el botón derecho, elija “Move” y selecciones el nombre del grupo al que desea mover este contacto:



DTU. Fig 5.

En caso de no estar seguros del "username" de la persona que vamos a añadir, podremos hacer una búsqueda por ejemplo, introduciendo el nombre:

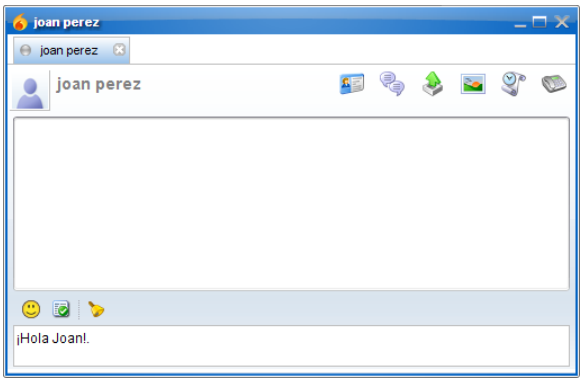


DTU. Fig 6.

Envío mensaje instantáneo:

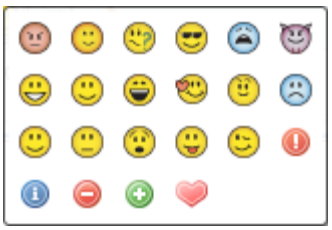
El envío de mensajes instantáneos es el método de contacto que más a menudo va a usar, además es fácil y rápido:

1. En su lista de contactos haga doble clic sobre el contacto con el que quiere chatear.
2. Escriba su mensaje:



DTU. Fig 7.

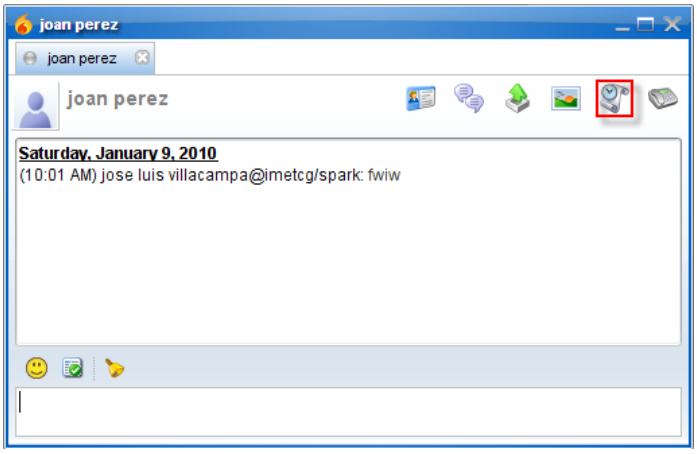
3. Para añadir un pequeño toque de colorido e informalidad en un mensaje, puede insertar un emoticono.
   Pulsando el botón:  se abrirá la lista de emoticonos disponibles:



DTU. Fig 8.

4. Cuanto este listo el mensajes se puede pulsar “enter” para enviar.

Se puede ver el histórico de mensajes enviados a alguien pulsando el botón de “View conversation history button”:



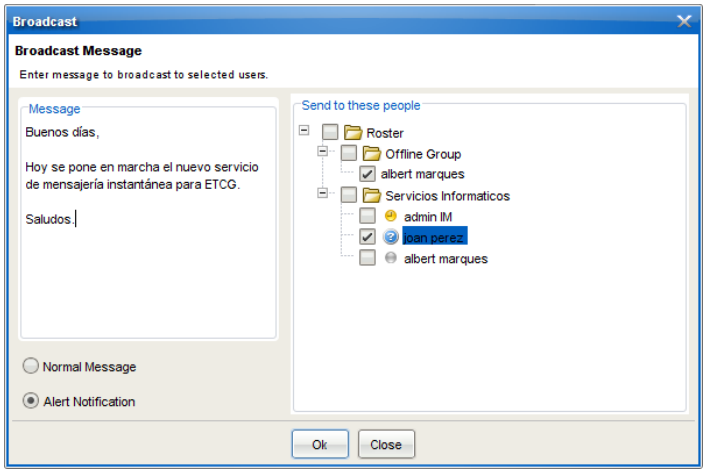
DTU. Fig 9.

Envío mensajes de Broadcast:

Cuando se tiene algo que decir a todos o varios de los contactos de tu lista, se puede emitir un mensaje de broadcast.
Para emitir estos mensajes se debe pulsar el botón de broadcast:

DTU. Fig 10.

Al pulsar este botón, se abrirá la siguiente ventana:

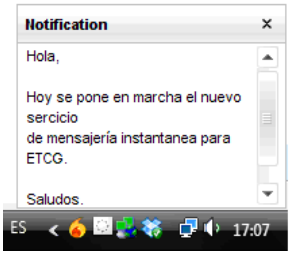


DTU. Fig 11.

En esta ventana se podrán hacer las siguientes selecciones respecto a los destinatarios del mensaje:

- Todo el rooster, se enviará el mensaje a todos los usuarios de todos los grupos.
- Todos los usuarios de un grupo concreto.
- Algunos usuarios de algún o algunos grupos.

Si el mensaje se envía como “Alert notification”, se presenta en forma de pop-up y aparece junto al reloj de Windows:



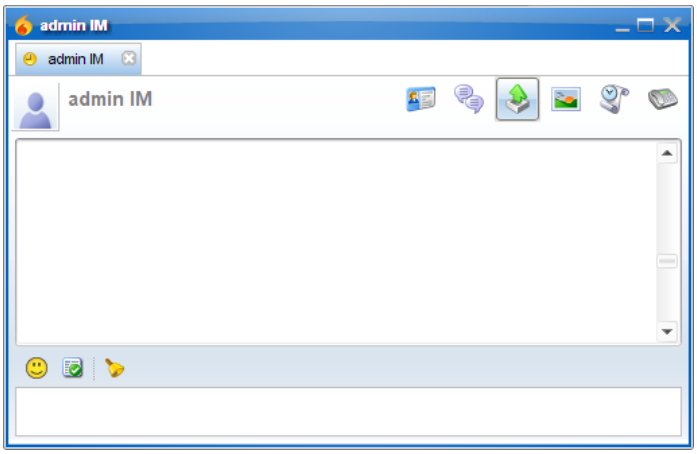
DTU. Fig 12.

Envío y recepción de ficheros:

El envío de ficheros a través de Spark es más rápido que si se hiciera a través del e-mail y además la principal ventaja es que inicialmente no esta limitado el tamaño máximo de los ficheros.

Para enviar un fichero seguiremos los siguientes pasos:

1. Abrir una sesión de chat con el usuario al que se le quiere transferir el fichero:



DTU. Fig 13.

2. Pulsaremos el botón de enviar ficheros:

DTU. Fig 14.

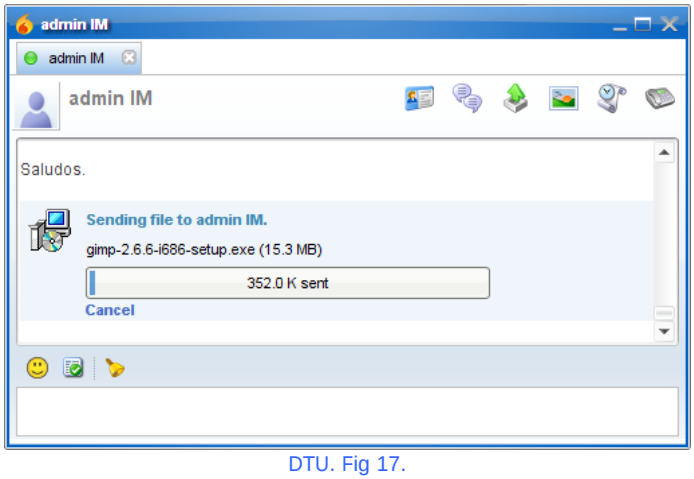
3. Navegamos con el explorador hasta encontrar el fichero y pulsamos abrir:

DTU. Fig 15.

4. El envío se queda waiting hasta que el usuario remoto acepte la transferencia del fichero:

DTU. Fig 16.

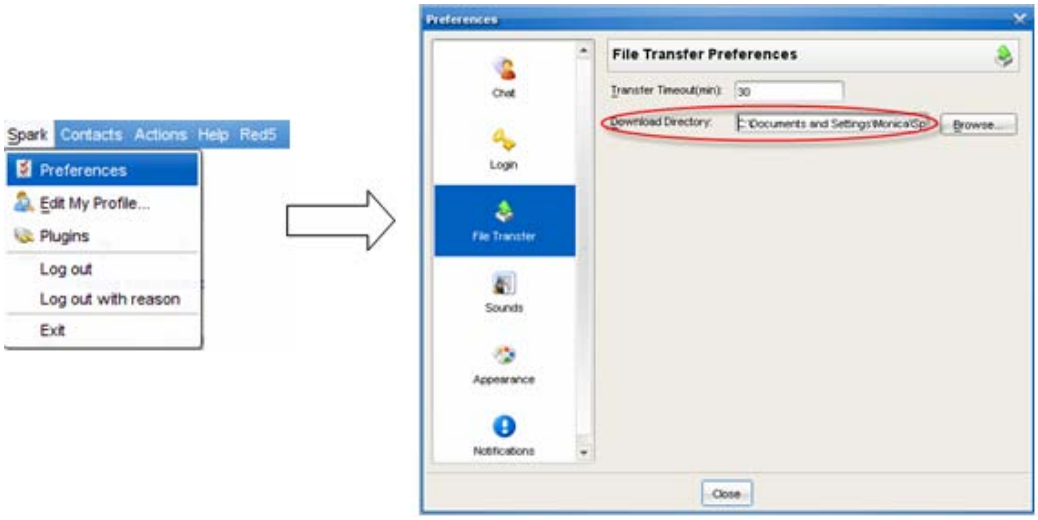
5. En el momento que se acepta se inicia la transferencia:



DTU. Fig 17.

También se puede iniciar una transferencia arrastrando el icono del fichero que queremos trasferir y dejándolo sobre el contacto de la lista al que le queremos enviar el fichero.

El path donde se ubican los ficheros transferidos se puede consulta y/o modificar accediendo desde el menú a: Spark, Preferences y File Transfer:

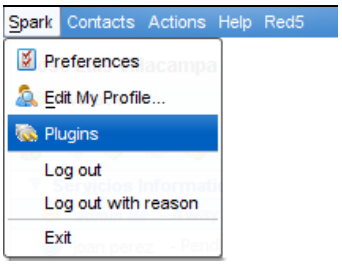


DTU. Fig 18.

Se podría limitar el tamaño de los ficheros instalando el plugin “Transfer Guard”.

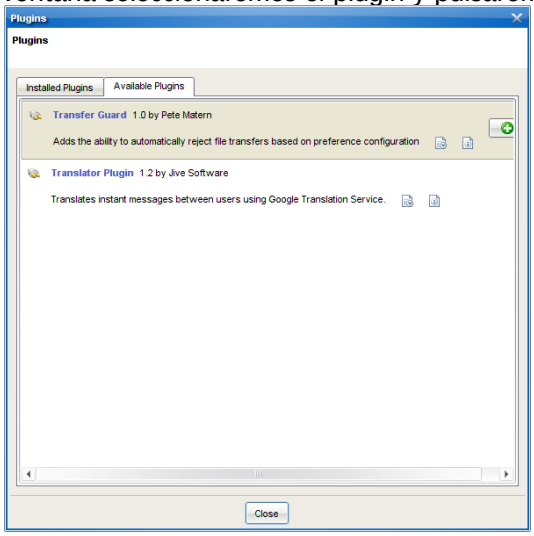
Para instalar esta plugin se deben seguir estos pasos:

1. Acceder desde el menú a: Spark, Plugins:



DTU. Fig 19.

2. Al abrirse esta ventana seleccionaremos el plugin y pulsaremos el botón: 

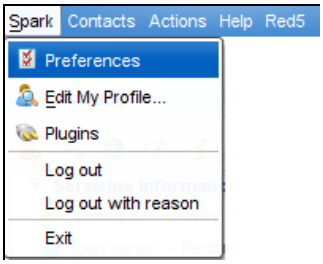


DTU. Fig 20.

3. Reiniciar Spark y quedará instalado el plugin.

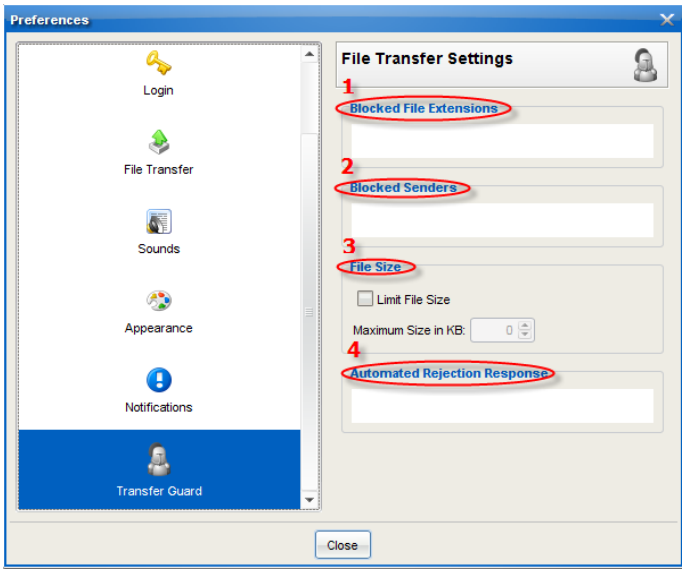
Para configurar las propiedades de las transferencias seguiremos los siguientes pasos:

1. Acceder desde el menú a: Spark, Prefences y Transfer Guard:



DTU. Fig 21.

2. Se podrán establecer las siguientes políticas de control en las transferencias de ficheros:



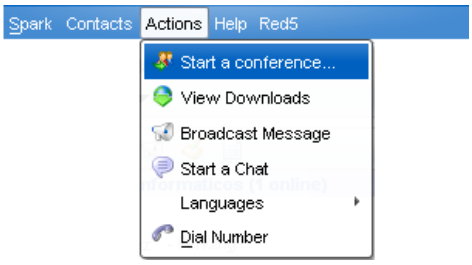
DTU. Fig 22.

Conferencias:

El uso de la conferencia es muy útil a la hora de compartir conversaciones entre varias personas.

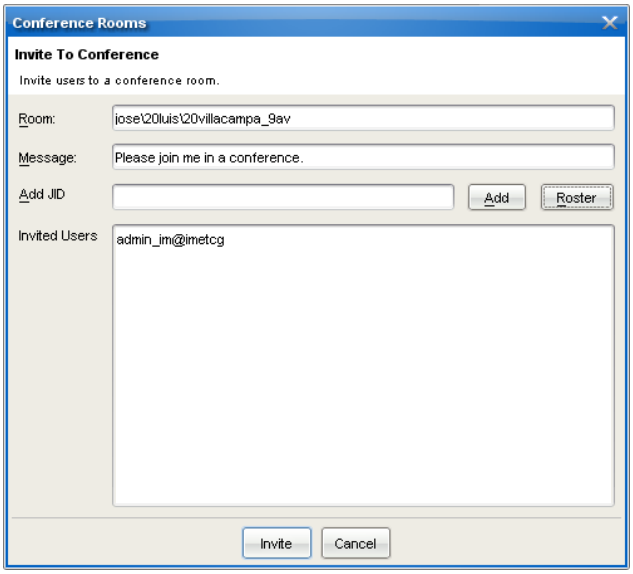
Para iniciar una conferencia se deben seguir estos pasos:

1. Acceder desde el menú a: Actions, Start a conference:



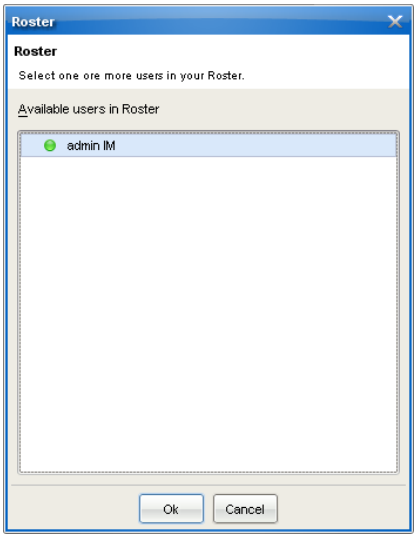
DTU. Fig 23.

2. Se abrirá la siguiente pantalla y haremos clic en el botón de “Roster”:



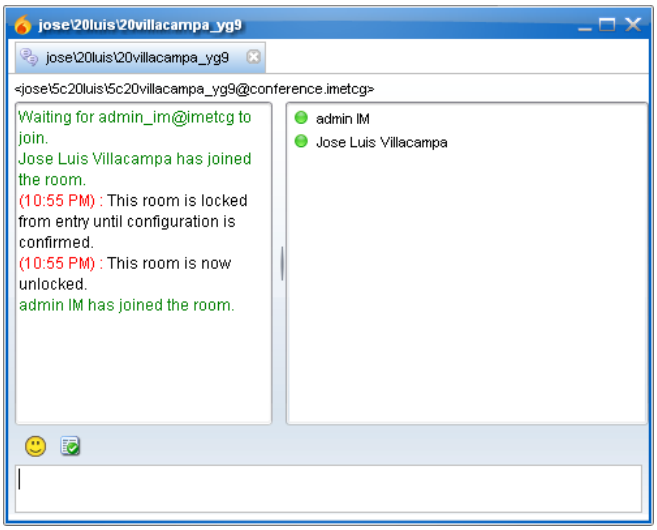
DTU. Fig 24.

3. Seleccionaremos los invitados a la conferencia y pulsaremos “OK”:



DTU. Fig 25.

4. Al finalizar la selección de invitados se debe pulsar botón de “Invite” y comenzará la conferencia. En la parte derecha de la ventana aparecerá la lista de conferenciantes y en la parte izquierda la sesión de chat:



DTU. Fig 26.

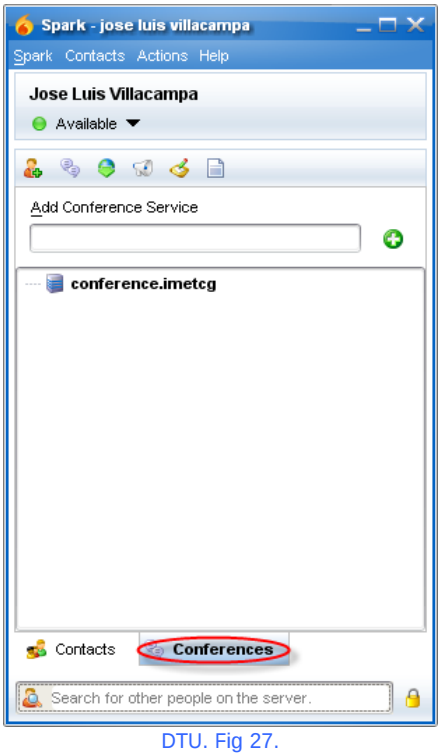
Salas de conferencia:

Las salas de conferencia son muy útiles a la hora de programar reuniones. Además, el uso de contraseña de acceso garantiza la privacidad de los conferenciantes.

Las salas de reuniones pueden ser temporales o privadas.

Los pasos a seguir para dar de alta una sala de conferencia son los siguientes:

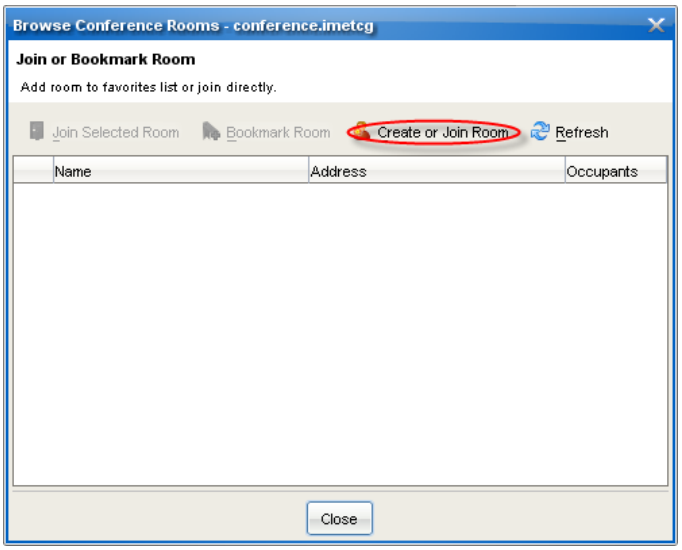
1. Hacer clic en la pestaña de conferencias:



DTU. Fig 27.

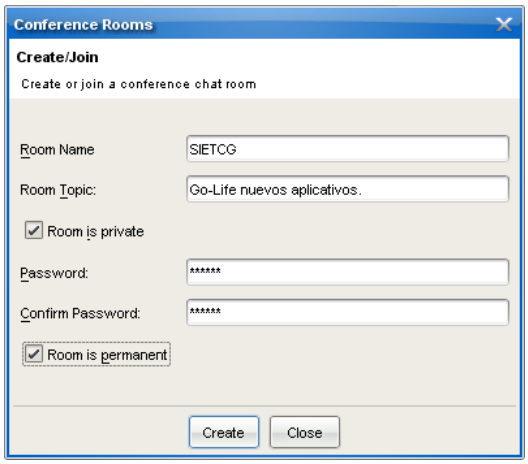
2. Hacer doble clic sobre el icono: conference.imetcg

3. Hacer clic sobre “Create or Join Room”



DTU. Fig 28.

4. Aparecerá esta ventana:



DTU. Fig 29.

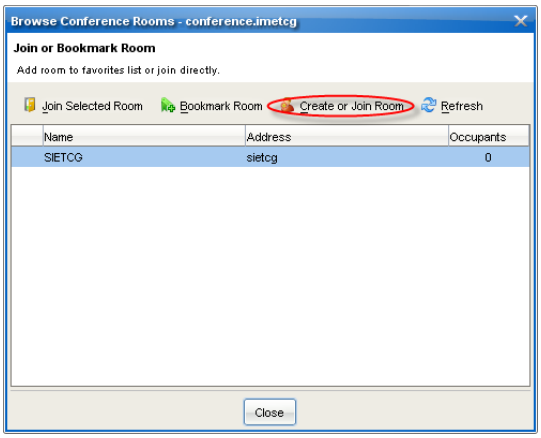
Si la sala de conferencia es privada se debe asignar un password que será necesario para poder entrar en la sala.

Si la sala es permanente y se desea borrar, se deberá solicitar a SIETCG para que la borren desde el aplicativo de administración.
Si la sala es temporal, tras el abandono del último conferenciante, se eliminará la sala.

Para unirse a esta sala de conferencias se deben seguir estos pasos:

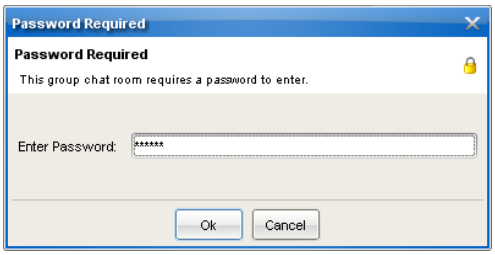
1. Bajo la pestaña de “conferences”, hacer doble clic en el icono: conference.imetcg

2. Seleccionamos la sala a la que nos queremos unir y pulsamos sobre el icono de “Create or Join Room”.



DTU. Fig 30.

3. Introducimos el password y ya estaremos unido a la sala:



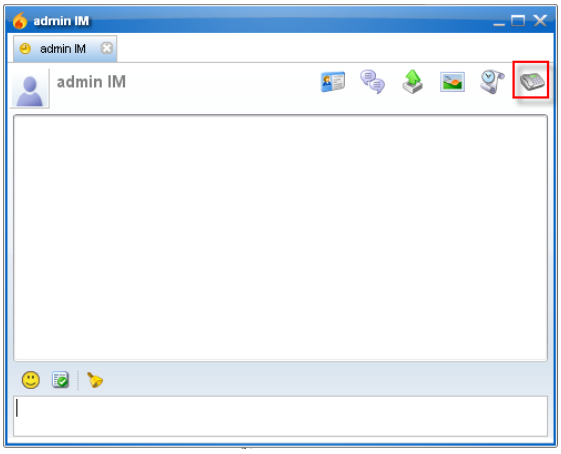
DTU. Fig 31.

Videollamadas:

Algunas cosas no se pueden expresar con mensajes de texto, para estas ocasiones cuando se necesita hablar con la persona con la que se está chateando, Spark proporciona los recursos necesarios para establecer una videollamada.

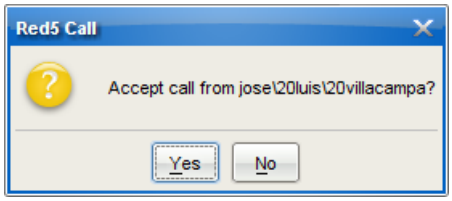
Los pasos a seguir para establecer una llamada de voz son los siguientes:

1. Hacer clic sobre el botón “Place a phone call”:



DTU. Fig 32.

2. En el otro extremo, a la persona que se está llamando le aparecerá este mensaje para confirmar el establecimiento de la llamada:



DTU. Fig 33.

3. Cuando se acepte la llamada quedará establecida la videollamada:

DTU. Fig 34.

Compartir escritorio:

Se podrá compartir el escritorio para que un usuario remoto de Spark pueda ver el contenido de su escritorio.

Para compartir un escritorio se requieren dos acciones:

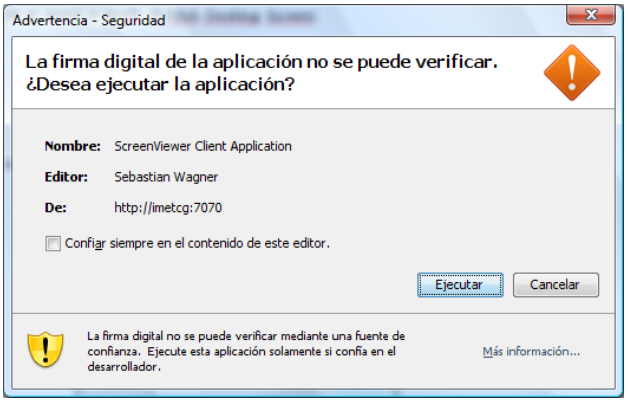
En el PC donde se va a compartir el escritorio:

1. Acceder desde el menú a: Red5, Publish Desktop Screen:

DTU. Fig 35.

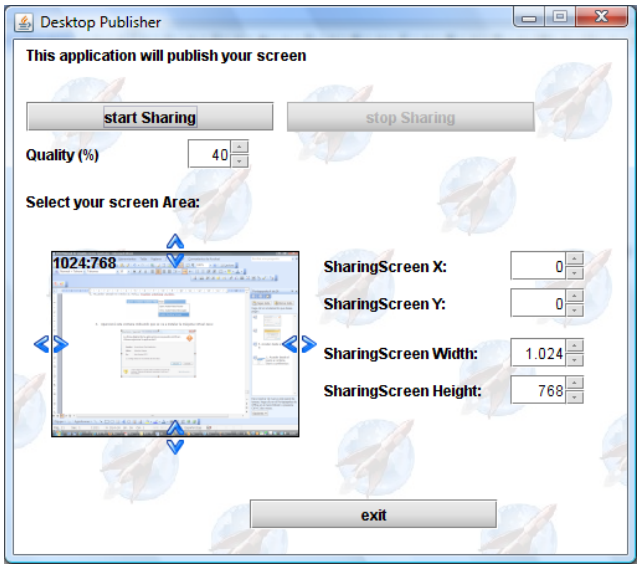
2. Se abrirá esta ventana y pulsaremos el botón de “Ejecutar”:

   Para el funcionamiento de esta funcionalidad es imprescindible tener instalada la máquina virtual java.



DTU. Fig 36.

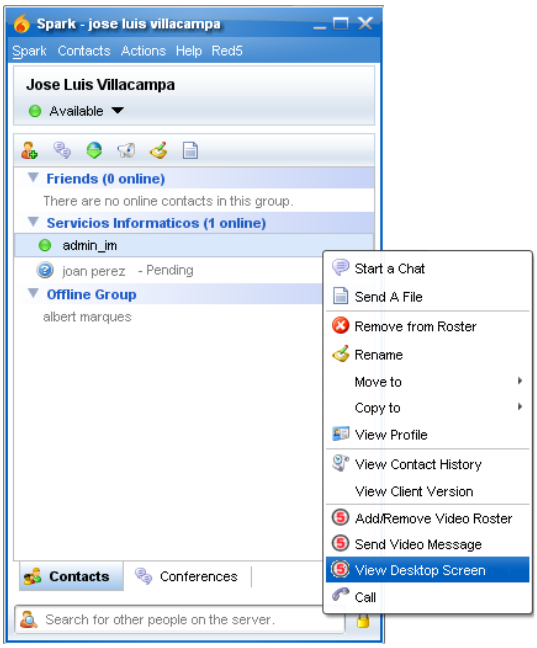
3. Pulsaremos el botón de “Start Sharing” para comenzar a compartir el escritorio:



DTU. Fig 37.

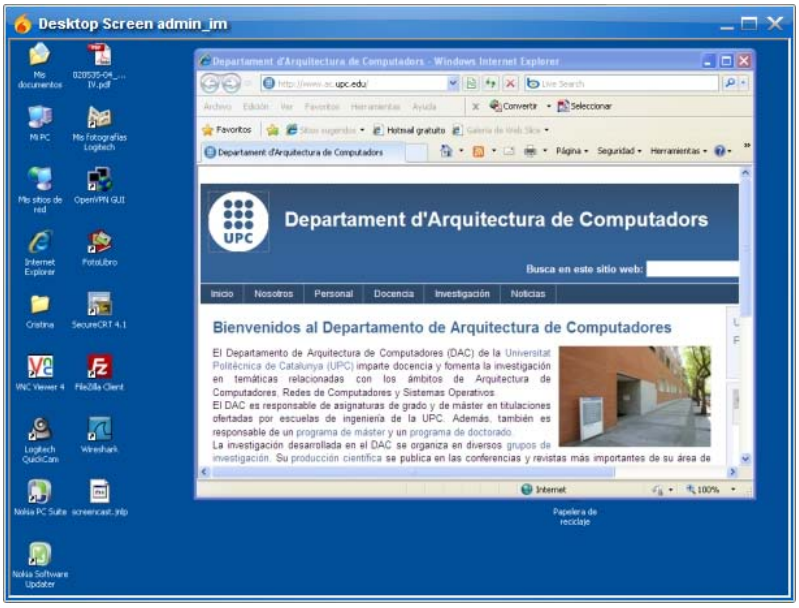
En el PC donde se va a ver el escritorio remoto:

1. Nos posicionaremos sobre el contacto de la lista, haremos clic con el botón derecho y después pulsaremos View Desktop Screen:



DTU. Fig 38.

2. Se abrirá una ventana en la que aparecerá publicado el escritorio del usuario remoto:



DTU. Fig 39.

DT.U.02. Manual de usuario cliente Sparkweb:

SparkWeb es el cliente web de mensajería instantánea. A través de este cliente podremos acceder al sistema de mensajería del departamento desde cualquier lugar y desde cualquier ordenador.

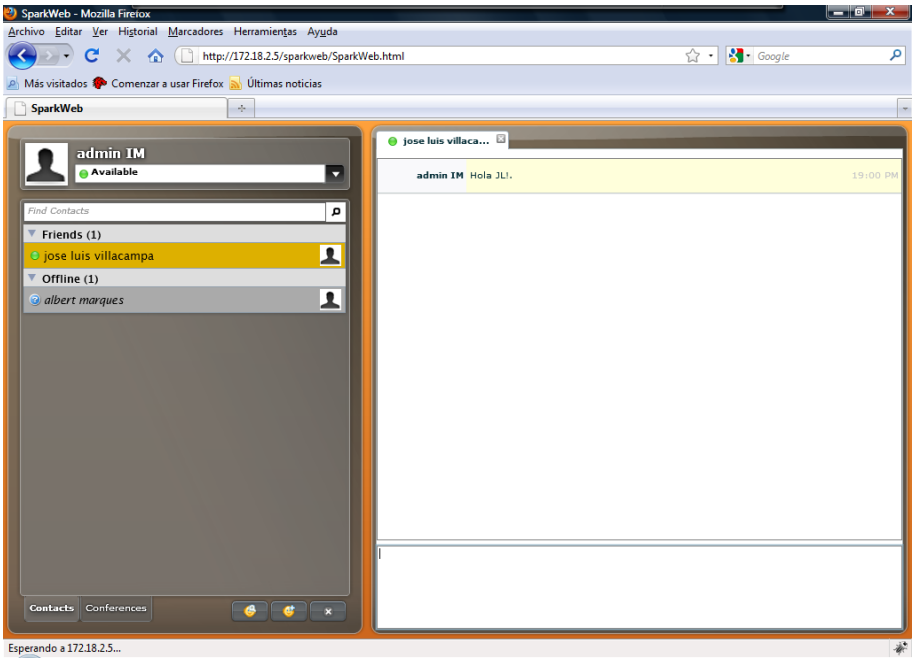
Conectarse con Spark:
La conexión con Sparkweb se hará desde un navegador web y se puede acceder a través de cualquiera de estos enlaces:
http://imetcg.upc.es/sparkweb/SparkWeb.html o http://imetcg.etcg.upc.es

Se abrirá una página web en la que serán requeridos los siguientes credenciales:

DTU. Fig 40.

Esta versión tiene un pequeño bug y si el usuario contiene espacios, cada espacio se debe sustituir por \20 que es el carácter ASCII que representa al espacio.

Conociendo Sparkweb:
El entorno no solo resultará familiar si has usado otros clientes de mensajería instantánea, sino que si se ha trabajado con el cliente Spark se podrá comprobar que el entorno es muy parecido.
Tras hacer "Login" nos aparecerá el siguiente entorno web:



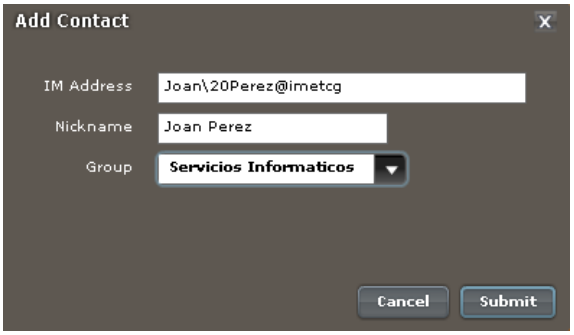
DTU. Fig 41.

Añadir contactos:

Deberás añadir los contactos con los que necesitas contactar habitualmente. Para ello se deben seguir los siguientes pasos:

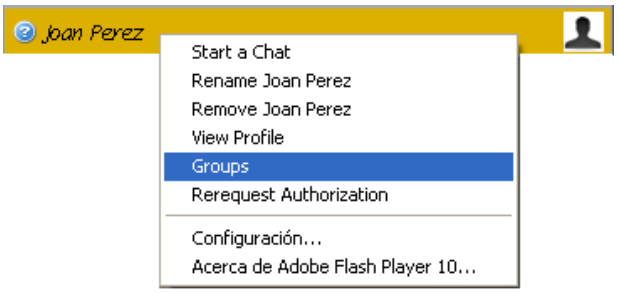
1. Haz clic en icono “Add Contact” : 

2. En la siguiente ventana introduciremos el nombre de la persona a añadir y pulsaremos el botón de Submit.
   Si el nombre contiene espacios también se deberán sustituir por el carácter ASCII \20.



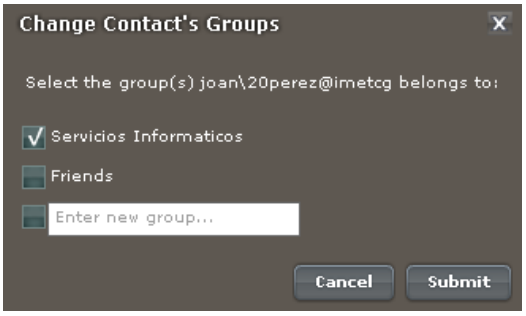
DTU. Fig 42.

Si se quiere mover de grupo a un contacto, se deberá hacer clic con el botón derecho sobre el contacto y seleccionar “groups”:



DTU. Fig 43.

Y seleccionaremos el grupo bajo el que se encontrará este contacto. Se puede seleccionar entre un grupo existente o se puede crear uno nuevo:



DTU. Fig 44.

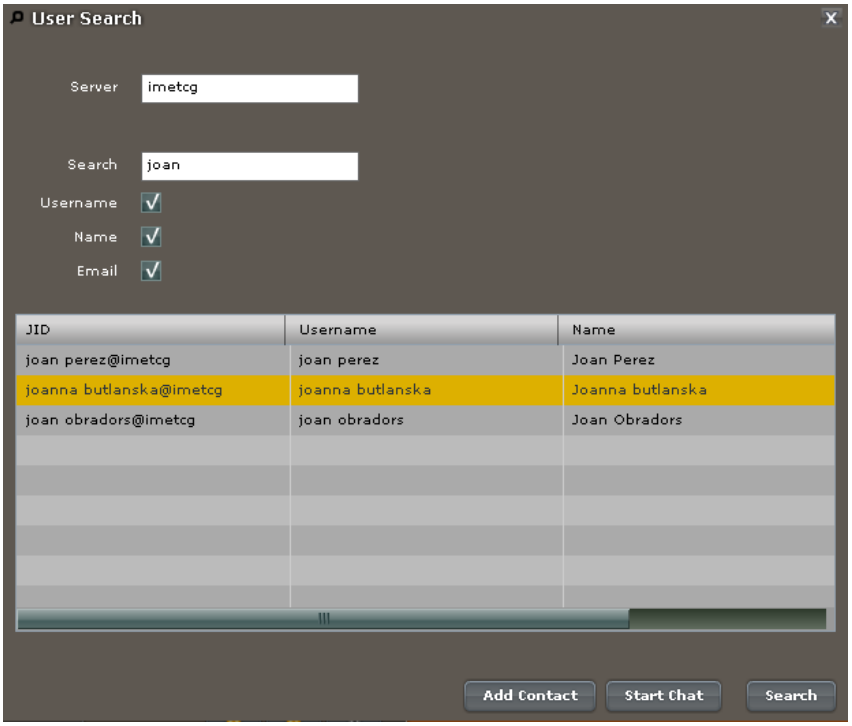
En caso de no conocer o no estar seguros del “username” de la persona que vamos a añadir, se podrá hacer una búsqueda por ejemplo, introduciendo el nombre.
Para hacer esta búsqueda seguiremos estos pasos:

1. Pulsar icono de “Search”: 

2. Introducir por ejemplo el nombre del contacto:

DTU. Fig 45.

3. Pulsar “Search” y aparecerán los resultados de la búsqueda:



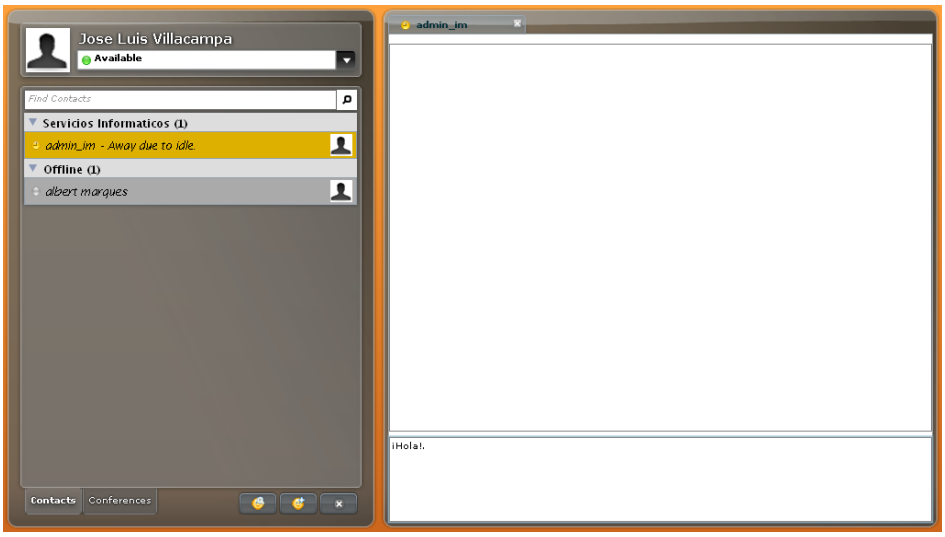
DTU. Fig 46.

4. Seleccionando el contacto y pulsando “Add Contact”, habrá quedado añadido el nuevo contacto.

Envío de mensajes instantáneos:
El envío de mensajes instantáneos es el método de contacto que más a menudo va a usar, además es fácil y rápido:

1. En su lista de contactos haga doble clic sobre el contacto con el que quiere chatear.

2. Escriba su mensaje:



DTU. Fig 47.

Salas de conferencia:

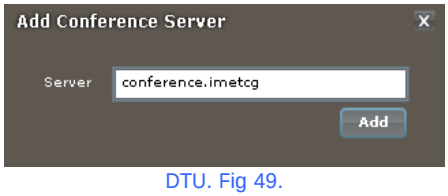
Antes de unirse o crear una sala de conferencias, se deberá dar de alta manualmente en el cliente web el servidor de conferencias, por lo que se seguirán estos pasos:

1. Hacer clic en la pestaña de conferencias:

DTU. Fig 48.

2. Pulsar el botón de “Add Server”: 

3. Añadiremos el servidor de conferencias. Este servidor se llama: conference.imetcg:

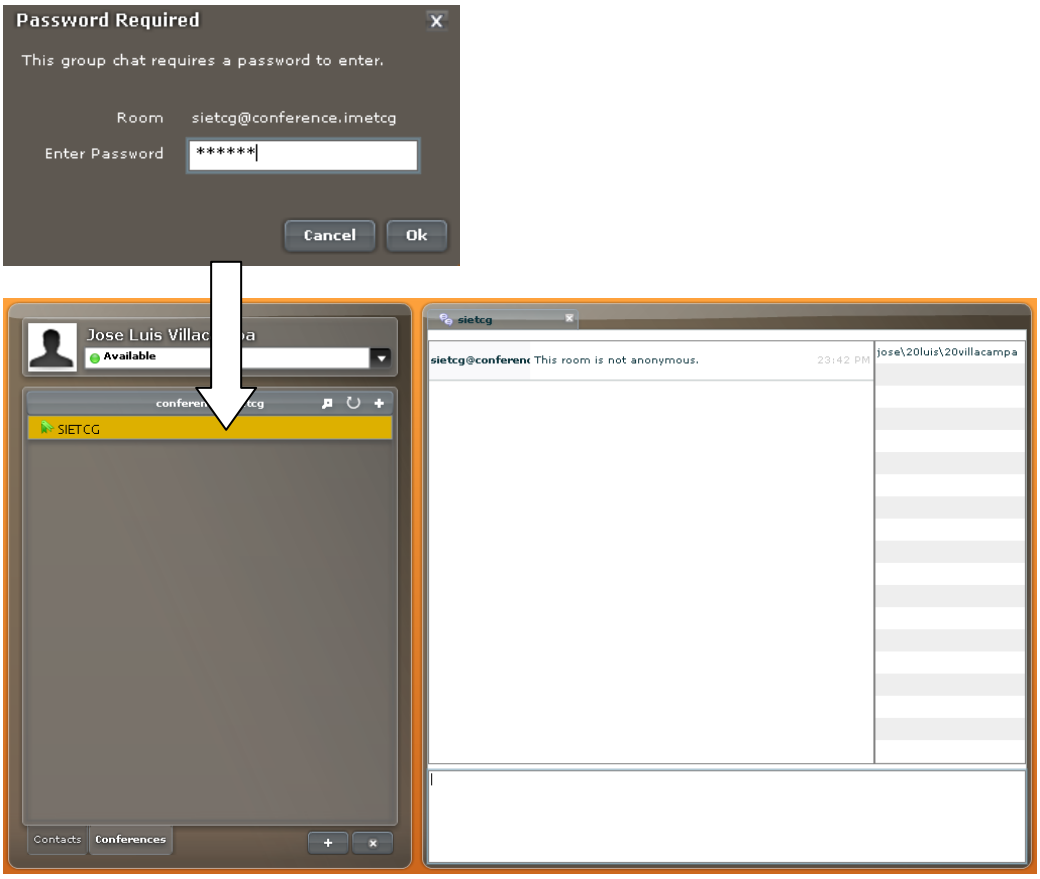


DTU. Fig 49.

4. Tras dar de alta el servidor de conferencias, aparecerán todas las salas dadas de alta:

DTU. Fig 50.

Una vez dado de alta el servidor de conferencias podremos unirnos haciendo doble clic sobre el nombre de la sala:



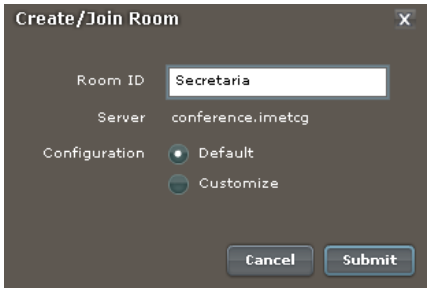
DTU. Fig 51.

También podremos crear una nueva sala, para esto seguiremos los siguientes pasos:

1. Pulsar el botón “Create Room”:

DTU. Fig 52.

2. Crearemos la sala de conferencia:



DTU. Fig 53.

Si se crea la sala eligiendo el modo “Default”, se creara una sala temporal.
Si se crea la sala eligiendo el modo “Customize”, podremos personalizar los siguientes parámetros:

DTU. Fig 54.

Departament d'Enginyeria del Terreny, Cartogràfica i Geofísica

Campus Diagonal Nord, Edifici D2.
C. Jordi Girona, 1-3. 08034
Barcelona

http://www.etcg.upc.edu

Guía rápida Sparkweb

Mensajería Instantánea

Departament d'Enginyeria del Terreny, Cartogràfica i Geofísica
UNIVERSITAT POLITÈCNICA DE CATALUNYA

SIETCG

Sparkweb:

Conectarse con Sparkweb:

Para abrir el cliente web de Spark, se puede acceder a uno de estos enlaces:

http://imetcg.upc.es/sparkweb/SparkWeb.html
http://imetcg.upc.es/

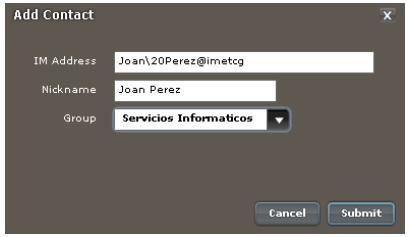
Introducir "Username" y "Password", serán los mismos credenciales que los utilizados para registrar el PC en el dominio.

Si el Username contiene espacios, los espacios deben sustituirse por el carácter ASCII: **\20**

Añadir contactos:

Hacer clic sobre el icono: 

Introducir el "Username" del contacto que se va a añadir y pulsar el botón de "Submit":

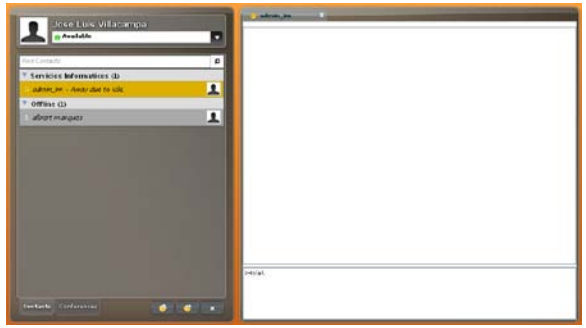


El nuevo contacto aparecerá en la lista de contactos y estará pendiente de recibir la confirmación por parte del contacto añadido.

Si el nombre contiene espacios también se deberán sustituir por el carácter ASCII \20.

Envío mensajes instantáneos:

En la lista de contactos, hacer doble clic sobre el contacto con el que se quiere iniciar el intercambio de mensajes instantáneos:



Salas de conferencia:

Alta servidor de conferencias:

1. Hacer clic en la pestaña de conferencias.
2. Pulsar el botón de "Add Server": 
3. Añadiremos el servidor de conferencias. Este servidor se llama: conference.imetcg:

4. Tras dar de alta el servidor de conferencias, aparecerán todas las salas dadas de alta:

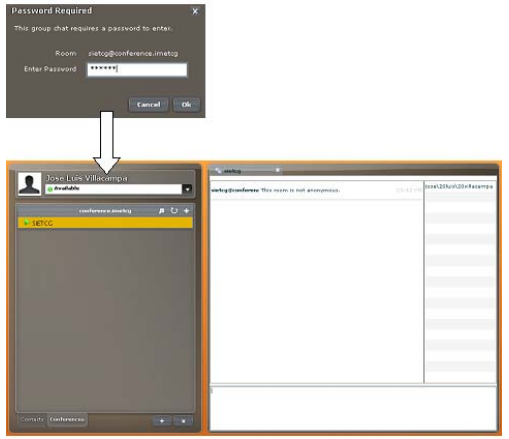
Crear salas de conferencia:

1. Pulsar el botón "Create Room":

2. Crear la nueva sala de conferencias:

Unión a salas de conferencia:

Podremos unirnos haciendo doble clic sobre el nombre de la sala e introduciendo el Password si lo requiere:



Video llamadas

Para establecer una video llamada se deben seguir estos pasos:

1. Desde una sisión de Chat, pulsar el botón de “Place a phone Call”:

2. En ambos extremos se deberá confirma el establecimiento de la video llamada y aceptar el acceso al micrófono y webcam del PC.

Compartir escritorio

Para compartir un escritorio se requieren dos acciones:

En el PC donde se va a compartir el escritorio:

1. Acceder desde el menú a: Red5, Publish.
2. Aparecera una ventana donde se ha de pulsar el botón de “Ejecutar”, se trata del cliente “ScreenViewer.
3. Pulsar el botón de “Start Sharing”

En el PC donde se va a ver el escritorio remoto:

Posicionarse sobre el contacto y haciendo clic con el botón derecho seleccionar “View Desktop Screen”.

Departament d'Enginyeria del Terreny, Cartogràfica i Geofísica

Campus Diagonal Nord, Edifici D2.
C. Jordi Girona, 1-3. 08034
Barcelona

http://www.etcg.upc.edu

Guía rápida Spark

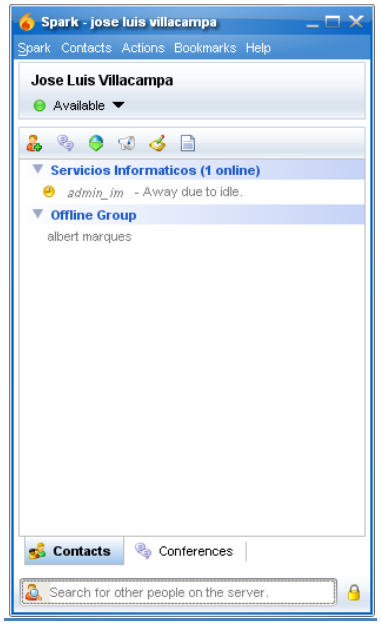
Mensajería instantánea

Departament d'Enginyeria del Terreny, Cartogràfica i Geofísica

UNIVERSITAT POLITÈCNICA DE CATALUNYA

SIETCG

Spark:



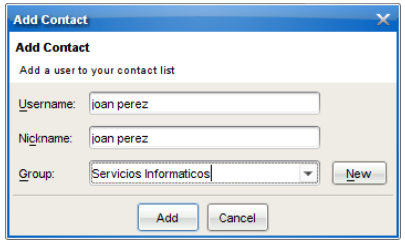
Conectarse con Spark:

Introducir "Username" y "Password", serán los mismos credenciales que los utilizados para registrar el PC en el dominio.

Añadir contactos:

Hacer clic sobre el icono "Add contact":

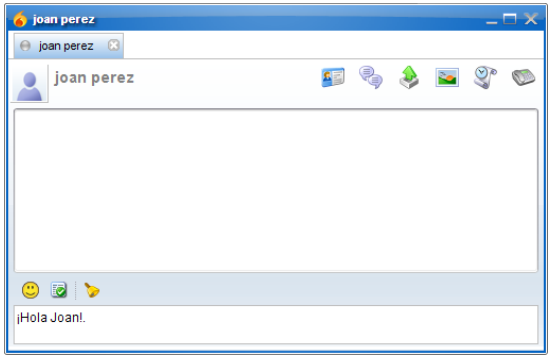
Introducir el "Username" del contacto que se va a añadir y pulsar el botón de "Add":



El nuevo contacto aparecerá en la lista de contactos y estará pendiente de recibir la confirmación por parte del contacto añadido.

Envío mensajes instantáneos:

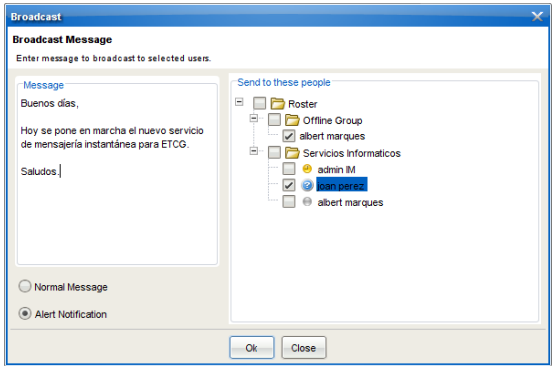
En la lista de contactos, hacer doble clic sobre el contacto con el que se quiere iniciar el intercambio de mensajes instantáneos:



Envío mensajes de broadcast:

Pulsar el botón de broadcast:

Seleccionar los destinatarios en la siguiente ventana:



Transferencia de ficheros:

Desde una sesión de chat, pulsar el botón de enviar ficheros:

Después se deberá explorar hasta encontrar el fichero y pulsar abrir.

Conferencias:

Para establecer una conferencia entre varios participantes se deben seguir estos pasos:

1. Acceder desde el menú a: Actions, Start a conference.
2. En la ventana que aparezca, hacer clic en el botón de "Roster" y seleccionar los usuarios a invitar a la conferencia.
3. Seleccionar los invitados y pulsar "OK".
4. Pulsar "Invite" para enviar las invitaciones de unión a la conferencia.

Salas de conferencia:

Crear salas de conferencia:

1. Hacer clic en la pestaña de conferencias.
2. Hacer doble clic sobre el icono:
   conference.imetcg
3. Hacer clic sobre "Create or Join Room".
4. Completar los datos de la nueva sala y pulsar el botón de "Create".

Unión a salas de conferencia:

1. Hacer clic en la pestaña de conferencias.
2. Hacer doble clic sobre el icono:
   conference.imetcg
3. Seleccionar la sala de conferencias y pulsar "Create or Join Room".
4. Introducir password si lo requiere.

Bibliografía y enlaces

Básicamente la principal fuente de información ha sido Internet, aunque también se ha recurrido a algún libro y a la documentación de alguna asignatura de ETSETB y otras fuentes docentes.

A continuación se detallan los documentos utilizados:

Enlaces de Internet:

- www.igniterealtime.org, es el portal de la comunidad de usuarios y desarrolladores de “Jive Software's open source Real Time Communications projects”. Son los desarrolladores de Openfire, Spark, Sparkweb y la mayoría de los plugins desarrollados para Openfire.

- http://osflash.org/red5, es el portal de la comunidad de desarrolladores del proyecto Red5, Open Source Flash Server.

- http://kraken.blathersource.org, es el portal de la comunidad de desarrolladores del proyecto Kraken XMPP IM Gateway.

- http://www.voztovoice.org, en este portal aparecen publicados varios artículos sobre soluciones Open Source para comunicaciones unificadas. De este portal se ha extraído una guía de instalación de Openfire y del cliente Sparkweb.

- http://knol.google.com/k/instalaci%C3%B3n-de-openldap-en-debian#, guía de instalación de plataforma OpenLdap.

- http://elrinconcito-carlos.blogspot.com/2008/10/mensajeria-interna-con-jabber-xmmp.html, articulo sobre mensajería instantánea Jabber.

- http://metajack.wordpress.com/2008/08/26/choosing-an-xmpp-server/, artículo sobre la comparativa de varios servidores XMPP.

- http://xmpp.org/software/servers.shtml, resumen de enlaces de la mayoría de portales de la comunidad de desarrolladores de servidores XMPP.

- http://www.voipforo.com/SIP/SIP_mensajes_error.php, cuadro resumen mensajes de error SIP.

- http://www.tribulinux.com/tutoriales-instalacion-completa-de-openser-132.html, portal web donde se publican artículos sobre software libre y se publica un articulo sobre como se instala Openser 1.3.2.

- http://www.voip.unam.mx/mediawiki/index.php/Instalaci%C3%B3n_B%C3%A1sica_OpenSER_/_OpenSIPS_/_Kamailio_con_Presencia, guía de instalación de Openser con Presencia.

- http://www.kamailio.org/docs/modules/1.4.x/, portal desarrolladores de Openser / Kamailio, de aquí se ha extraído la descripción y configuración de los módulos que tiene disponibles.

- http://www.jabberes.org/, portal web de la mensajería instantánea libre Jabber.

Bibliografía:

- "Openfire Administration", Mayank Sharma, Packt Publishing, ISBN 978-1-847195-2b-5.
- "Internet Communications using SIP delivering VoIP and Multimedia Services with Session Initiation Protocol", Henry Sinnreich , Alan B. Johnston . Jul.2006 eBook-DDU.
- "XMPP : the definitive guide : building real-time applications with jabber Technologies", Saint-Andre, Peter; Smith, Kevin; Tronçon, Remko Book / 2009.
- Programming Jabber, Adams, D.J Book / 2001.
- "Estudio de viabilidad telefonía IP en red inalámbrica UPC", Jerez Álvarez, Cristina. Proyecto final de carrera, alojado en 'upcommons.upc.edu/pfc/handle/2099.1/7312'.
- "Implantación de un sistema VoIP basado en Asterisk", Barberán Plaza, Javier. Proyecto final de carrera, alojado en 'http://upcommons.upc.edu/pfc/handle/2099.1/6798'.

Otra documentación:

- "Metodología de gestión de proyectos TIC", Lamarca, Ignacio y Rodríguez, José Ramón. Apuntes master de dirección y gestión de sistemas y tecnologías de la información
- "Redes, sistemas y servicios avanzados de telecomunicación". ETSETB, apuntes asignatura optativa XSSAT.